Universidade de Aveiro
2009

Departamento de Electrónica, Telecomunicações e Informática

João Filipe Amaral Costa

Controlo dos Movimentos de um Robot de Seis Eixos

João Filipe Amaral Costa

Controlo dos Movimentos de um Robot de Seis Eixos

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações (Mestrado Integrado), realizada sob a orientação científica do Dr. António Ferreira Pereira de Melo, Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e Dr. Telmo Reis Cunha, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

O júri

Presidente

Professor Doutor José Carlos Esteves Duarte Pedro

Professor Catedrático do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro

Vogais

Professor Doutor Jaime dos Santos Cardoso

Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor António Ferreira Pereira de Melo

Professor Catedrático do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro (Orientador)

Professor Doutor Telmo Reis Cunha

Professor Auxiliar do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro (Co - Orientador)

Agradecimentos

Ao Professor Catedrático António Ferreira Pereira de Melo, e Professor Auxiliar Telmo Reis Cunha um agradecimento especial pela sua orientação e conhecimentos transmitidos ao longo da minha dissertação.

Quero agradecer ao meu pai e à minha mãe, Joaquim e Lurdes, por todo o esforço que fizeram para que eu concluísse com êxito o meu curso.

Agradeço também à minha namorada, Rita, pelo apoio e carinho prestado ao longo da minha formação académica.

Um agradecimento também ao meu grande amigo Daniel que compartilhou comigo tantas tardes e noites de estudo e actividades de carácter lúdico.

Por fim, mas não menos importante, agradeço aos meus familiares, amigos e colegas pela entreajuda e amizade, permitindo-me superar algumas etapas mais complicadas.

Palavras-chave

Robot, Manipulador industrial, Cinemática de um manipulador, *Labview*, Sistemas de controlo, Redes de comunicação em ambiente industrial.

Resumo

No mundo actual onde os processos de fabrico industrial são cada vez mais complexos e de menor duração é colocado um desafio de enorme aos empresários de todo o mundo, como produzir mais e melhor. Para responder a esse desafio, o mundo industrial virou-se para a máquina em busca da solução. Com o desenvolvimento de novas tecnologias, e o decréscimo dos preços de um manipulador industrial, este tem vindo a assumir um papel crescente no meio industrial. As unidades fabris dos dias de hoje, são completamente diferentes das existentes à vinte anos atrás hoje em dia uma fábrica é já uma verdadeira rede informática, como tal existe a necessidade de integração toda a maquinaria nessa rede, para que a sua produtividade cresça com o menor custo possível.

O presente trabalho visa contribuir para a solução do desafio que é colocado nos dias de hoje ao mundo empresarial. O seu desenvolvimento ocorreu numa parceria entre o Departamento de Electrónica, Telecomunicações e Informática e a empresa Selmatron sedeada em Aveiro.

O projecto desenvolvido visava a construção de uma aplicação informática capaz de controlar um manipulador industrial. A linguagem de programação utilizada para a programação da aplicação informática foi o *Labview*.

Seguidamente procedeu-se ao desenvolvimento da aplicação de software de interface com o manipulador, com diversas funcionalidades que permitem ao utilizador lidar com o manipulador de uma forma muito simples, uma vez que é a própria aplicação que efectua o controlo dos actuadores do manipulador.

Elaborou-se, também, uma aplicação de software capaz de simular o comportamento de um manipulador real modelo PUMA 760 (idêntico ao manipulador real), permitindo assim desenvolver a aplicação de interface de uma forma independente do hardware em análise.

Keywords Robot, Industrial Manipulator, Manipulator Kinematics, *Labview*, Control systems, Communication networks in industrial environment.

Abstract

In the actual world, where the industrial making processes are every time more complex and short duration time, a challenge is placed before the worldwide businessmen, like to produce more and better. In order to respond to this challenge, the virtual world came to the machine, looking for a solution. Along with the new technologies development, and the decreasing costs of industrial handlers, these ones have been assuming an increasing role in the industrial environment. Nowadays factories are completely different from those that existed 20 years ago, actually they are a truly informatics network, where there's a full machinery integration need in that network, so that the productivity grows with the least possible cost.

The current work aims to contribute to the challenge placed nowadays in the enterprise world. Its development occurred with a partnership between the Electronics, Telecommunications and Informatics Department of Aveiro's University and the Selmatron company, settled in Aveiro.

The developed project aimed to build a computer application which is able to control an industrial manipulator. The programming language used to the computer application was the *Labview*.

Afterwards, it was proceeded to the application software handler interface development, containing several features that allow the user to handle in a very simple way, since is the application itself that controls the handler's actuators.

It was also designed a software application which is able to simulate the behavior of the real handler PUMA 760 (identical to the real handler), thus it was possible to develop the interface in an independent way from the analysed hardware.

Índice de Conteúdos

Índice de Figuras

Índice de Gráficos

Índice de Tabelas

Acrónimos

ADS - Automation Device Specification

IP – *Internet Protocol*

NI – *National Instruments*

OPC – OLE for *Process Control*

OLE – *Object linking and Embedding*

PDO – *Process Data Object*

TCP – Transmission Control Protocol

V – Volt

VI – Virtual Instrument

Capítulo 1

1. Introdução

1.1. Enquadramento

Nos dias de hoje o mundo tornou-se uma "aldeia global", onde tudo está à distância de um simples "clique", existindo assim uma necessidade crescente no mundo empresarial de tirar o máximo rendimento de todos os recursos disponíveis a uma empresa. Esses recursos podem ser humanos, materiais e também monetários. Assim, uma empresa deve ser capaz de gerir, o melhor possível, todos os recursos ao seu dispor, uma vez que, como tudo na vida, estes são finitos. Esta gestão eficaz dos recursos de uma empresa é uma necessidade face ao seu objectivo primordial de maximizar os seus lucros.

O meio industrial pode seguir caminhos distintos para a maximização de lucros, pode aumentar o preço dos seus produtos ou serviços, pode tentar conquistar novos mercados para os seus produtos, ou pode diminuir os custos de produção mantendo, ou mesmo baixando, o preço dos seus produtos ou serviços. O terceiro caminho, é o que normalmente é o escolhido pelo mundo empresarial. Quando tal acontece o empresário recorre à máquina para substituir o homem, uma vez que esta produz se necessário ininterruptamente, não tem reivindicações salariais podendo ser desmontada e levada para qualquer ponto do globo onde continuará a produzir. Foi neste contexto que surgiu a Robótica Industrial, surgindo associado a esta os robots manipuladores industriais.

Actualmente o robot está disseminado um pouco por todo o mundo, estando o seu uso generalizado. Nos dias de hoje o robot é usado em linhas de montagem de um sem número de indústrias. Um exemplo da sua aplicação é a indústria automóvel, onde é usado para soldar, montar ou mesmo pintar. Outro exemplo é a indústria farmacêutica onde é usado para colocar tubos de ensaio nos instrumentos de medida. O

aparecimento do robot não se deve só à necessidade de redução de custos de produção mas também, à necessidade crescente de mobilidade e versatilidade das empresas, bem como à necessidade de "alguém" que efectuasse trabalhos repetitivos, tediosos, e de grande grau de perigosidade. Um exemplo deste último aspecto é o uso do robot para a investigação do fundo dos oceanos, em missões espaciais e também em centrais nucleares.

Por todos os motivos apresentados, e devido ao grande avanço tecnológico existente, o estudo deste tipo de dispostivos electrónicos reveste-se de grande importância. Assim nesta dissertação pretende-se estudar com detalhe todos os elementos constituintes de um braço robótico e apresentar um algoritmo de controlo capaz de fazer com que este efectue uma determinada tarefa. Todo o algoritmo de controlo será desenvolvido na linguagem de programação gráfica *Labview*.

Toda a dissertação foi desenvolvida em parceria com a empresa Selmatron, sendo esta empresa a detentora de todo o material utilizado. A Selmatron, sendo uma jovem empresa, possui já um conhecimento e experiência bastante alargada na área da automação e electrónica industrial, sendo estas as suas principais áreas de negócio. Por último falta apenas referir que a dissertação foi desenvolvida nas instalações da referida empresa, uma vez que era lá que se encontrava todo o material necessário.

1.2.Objectivos

Com a realização desta dissertação pretende-se criar uma aplicação informática capaz de controlar um braço robótico de seis Eixos. Toda a aplicação será desenvolvida em *Labview*, usando para o efeito as várias valências e potencialidades que esta ferramenta dispõe. Todo o algoritmo desenvolvido em *Labview* terá que implementar no braço robótico a capacidade de este efectuar tarefas sobre trajectórias pré definidas e/ou definidas em tempo-real.

Adicionalmente ao objectivo principal desta dissertação, pretende-se também efectuar o estudo de uma gama alargada de áreas a este associadas, tais como o estudo da cinemática de manipuladores, dos protocolos de comunicações tipicamente

utilizados, da electrónica de interface aos actuadores e sensores, entre outras. Será dado algum relevo à parte de comunicações nesta dissertação, uma vez que constitui uma parte fulcral da indústria da robótica. Esta dissertação também abordará a temática dos dispositivos electrónicos usados tipicamente em aplicações deste tipo. Por fim toda a teoria de controlo também será objecto de estudo e reflexão ao longo da dissertação.

1.3. Organização do documento

Esta dissertação está estruturada em 6 capítulos, de acordo com a seguinte disposição:

No capítulo dois é feita uma apresentação de toda a história e evolução do robot. São também apresentados os vários tipos de manipuladores industriais existentes nos dias de hoje em muitas unidades fabris. Por fim é feita uma análise do estado da arte e apresentada a solução que será adoptada.

O capítulo três apresenta todos os fundamentos teóricos necessários para um melhor entendimento da dissertação. São apresentados conceitos como a localização de um ponto num espaço tridimensional, bem como a cinemática de manipuladores.

O capítulo quatro descreve as várias fases de desenvolvimento desta dissertação. Neste capítulo é também referido todo o *hardware* utilizado para a realização da dissertação.

O capítulo cinco apresenta a aplicação informática desenvolvida para simular o comportamento de um manipulador modelo PUMA 760. São apresentados também todos os procedimentos necessários para executar de forma correcta o simulador desenvolvido.

Por fim, no sexto capítulo, são apresentadas as conclusões da dissertação, bem como o trabalho futuro a desenvolver numa fase posterior.

Capítulo 2

2. Introdução Teórica e Estado da Arte

2.1.Estado da Arte

A moderna indústria da robótica industrial dos dias de hoje teve que percorrer um longo caminho, e por vezes sinuoso, até atingir o grau de desenvolvimento actual. O termo robot tem origem na palavra checa *robota* que significa trabalho, tendo sido usado pela primeira vez em 1921 pelo escritor Karel Capek no seu romance "Rossum's Universal Robots". Neste romance o escritor checo idealizava que um robot era uma máquina de aspecto semelhante ao dos humanos e que seria uma máquina de trabalho incansável com características fantásticas, mesmo para os padrões tecnológicos dos dias de hoje. Ao longo dos anos a ficção científica foi transformado o robot numa máquina absolutamente fantástica, podendo por vezes até ter sentimentos e capacidade para os exprimir, isso mesmo está bem patente na série "Star Wars" ou no famoso filme "I Robot".

Tal como na ficção científica, onde o robot simboliza o trabalhador ideal utilizado para realizar tarefas repetitivas e desprestigiantes para os "humanos", o homem sempre teve o sonho, e a necessidade, de se reproduzir a si próprio por meios mecânicos criando assim o seu escravo ideal, obediente, insensível, descartável e de baixo custo. Os primeiros registos que existem de trabalhos nessa área remontam ao antigo povo egípcio, que colocava braços mecânicos nas estátuas dos seus deuses. [1] Esses mesmos braços mecânicos eram operados por sacerdotes, dando desta maneira a ilusão aos seus fiéis que eram movidos por inspiração divina. Enquanto que o povo egípcio usava braços mecânicos em estátuas para simular inspiração divina, o povo da antiguidade grega desenvolveu relógios de água com figuras para ilustrar a ciência da hidráulica. [2]

Tendo já visto um pouco da origem do robot falta agora encontrar uma definição para este dispositivo. Se se procurar em qualquer livro que aborde esta temática observa-se que existem enumeras definições para o que é um robot, sendo que praticamente cada actor tem a sua definição. Em seguida vai ser apresentada a definição do robot *Institute of America*:

"*A programmable multi–function manipulator designed to move material, parts, or specialized devices through variable programmed motions for the performance of a variety of tasks".*

(Schlussel,1985)

Tal como já atrás foi referido, o homem sempre sentiu necessidade de se representar por meios mecânicos, como desenvolver todo um sistema mecânico que emulasse o seu corpo seria extremamente complexo, desenvolveu um dispositivo que substituísse apenas o seu braço, concebeu assim o manipulador ou robot industrial. O manipulador industrial é constituído por juntas e por elos. Os elos são normalmente blocos alongados e rígidos, ligados em série através das juntas, blocos rotativos que fazem com os elos variem a sua posição relativa ao longo do movimento. Dependendo da aplicação, a combinação entre elos e juntas pode variar.

Genericamente o braço humano é constituído por 3 partes rotativas, ou juntas, e por 2 rígidas, ou elos. A primeira junta existente no braço é o ombro, tendo esta três graus de liberdade, a segunda junta é o cotovelo que tem apenas dois grau de liberdade e finalmente o punho que tal como o cotovelo possui dois graus de liberdade. Convém salientar que se entende por grau de liberdade de uma junta, o número total de movimentos independentes que um dispositivo pode efectuar. O ombro e o cotovelo são ligados através do Úmero que funciona como o elo, enquanto que o cotovelo e punho são ligados através do Rádio e Cúbito, segundo elo. Somando os graus de liberdade de todas as juntas que constituem o braço humano concluísse que este tem sete graus de liberdade.

Actualmente a maioria dos manipuladores usados na indústria são em tudo semelhante ao braço humano, contudo têm apenas seis graus de liberdade. O braço

humano com sete graus de liberdade é totalmente redundante, então qualquer ponto de uma qualquer trajectória pode ser atingido de diferentes maneiras, quando tal acontece diz-se que não existem configurações singulares. Os manipuladores industriais dos dias de hoje têm apenas seis graus de liberdade, sendo também eles redundantes, apresentando contudo algumas configurações singulares, os seus seis graus de liberdade garantem que todos os pontos do espaço de trabalho do manipulador estão acessíveis. Para que o manipulador seja totalmente controlado é necessário que o programador tenha conhecimento das suas singularidades, se tal acontecer este pode evitar trajectórias que contenham as singularidades, mantendo o braço robótico sempre controlado.

Os manipuladores indústrias actuais podem ser divididos em duas partes distintas: o braço e o punho. O braço é constituído pelas primeiras juntas, sendo estas que determinam a localização do elemento terminal ou garra, enquanto o punho determina a orientação espacial da garra. Para melhor se entender este conceito pode-se recorrer novamente ao exemplo do braço humano, sendo a sua localização dada pelo ombro e cotovelo e a orientação da mão dada pelo punho. Em robots que tenham seis graus de liberdade o grupo de juntas que constitui o braço do manipulador é formado pelas três juntas mais próximas da base, sendo esse grupo usualmente designado de juntas principais, o grupo de juntas que constitui o punho, é formado pelas três restantes juntas do manipulador sendo designadas por juntas secundárias ou juntas do punho. [1]

Como já foi referido, o conjunto das juntas que se seguem ao braço do manipulador são denominadas de punho, tendo como responsabilidade a orientação espacial da garra. A garra é orientada segundo um referencial de três eixos. Aos ângulos de rotação em torno de cada um dos eixos dá-se normalmente o nome de ângulos de Euler, sendo que o seu significado depende da combinação de rotações usadas. Ao todo existe um total de 12 combinações distintas, contudo na robótica as mais comuns são a *Roll – Pitch – Yaw* (ZYX), e o Tipo II (ZYZ) dos ângulos de Euler, também conhecido pelo nome *Roll – Pitch - Roll*, também denominado por punho esférico. [3] O primeiro tipo de configuração é semelhante ao punho humano, contudo a configuração mais usada na robótica de manipulação é a segunda, devido a esta

apresentar maior simplicidade. [3] Convém alertar para o facto de este tipo de configuração introduzir singularidades no manipulador, como tal é necessário que o programador seja alertado para as mesmas, a fim de no planeamento da trajectória as evitar, mantendo assim o manipulador controlado.

Até este ponto foi descrito o robot manipulador industrial como sendo uma implementação mecânica do braço humano, o chamado braço articulado ou robótico, contudo este tipo de robots possuem mais configurações além da apresentada. A classificação deste tipo de dispositivo varia consoante a combinação de juntas (cintura, ombro e cotovelo), que são usadas na sua construção. Tipicamente existem dois tipos de juntas: as rotacionais onde o movimento relativo dos elos é rotacional, e as prismáticas onde o movimento relativo dos elos é linear. Os robots manipuladores industriais podem assim ser agrupados segundo a sua geometria em cinco classes, cujas principais características são de seguida apresentadas.

A primeira classe, a **Cartesiana** (**PPP**), é a mais simples das cinco classes. Os manipuladores pertencentes a esta classe usam exclusivamente juntas prismáticas, e têm a forma da Figura 1. Esta classe tem as seguintes vantagens: apresenta um espaço de trabalho rectangular estando todos os pontos acessíveis; tem movimentos lineares nas três dimensões; tem um modelo cinemático simples; é de estrutura rígida; apresenta a possibilidade de integração de *drive*s pneumáticos de baixo custo para tarefas *pick and place*. Contudo tem as seguintes desvantagens: tem grandes dimensões; apresenta a impossibilidade de acesso a objectos que estejam localizados sob outros. Este tipo de geometria é tipicamente utilizado em manipuladores que efectuem tarefas de transporte de materiais de grande peso a uma curta distância, sendo bastante utilizados na indústria metalúrgica e de construção naval. [2]

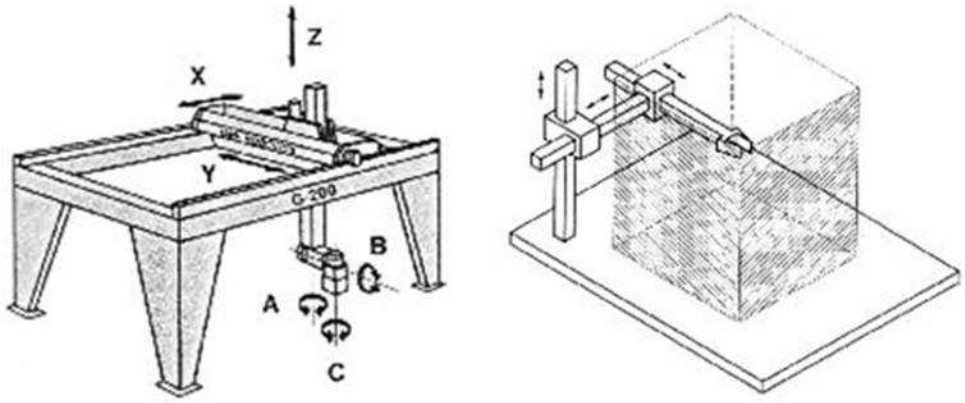

FIGURA 1 - MANIPULADOR DE GEOMETRIA CARTESIANA

A classe dos robots manipuladores industriais **Cilíndrica** (**RPP**), visível na Figura 2, é em tudo semelhante á anterior, contudo nesta classe a junta da cintura, ou base, é rotacional, sendo as restantes prismáticas. No que toca ao espaço de trabalho, este apresenta a forma de um cilindro graças à combinação de rotação com translação, estando desta maneira todos os pontos acessíveis. Tal como a classe anterior esta apresenta vantagens e desvantagens na sua utilização, que seguidamente são enumeradas. Vantagens: modelo de cinemático simples; de fácil visualização; boa acessibilidade a pequenas cavidades. Desvantagens: espaço de trabalho restrito; as juntas prismáticas têm bastante dificuldade em impedir a entrada de pó e também de líquidos, podendo estes afectar o normal funcionamento da junta. Os robots desta classe são tipicamente usados para o transporte de materiais leves a curtas distâncias.

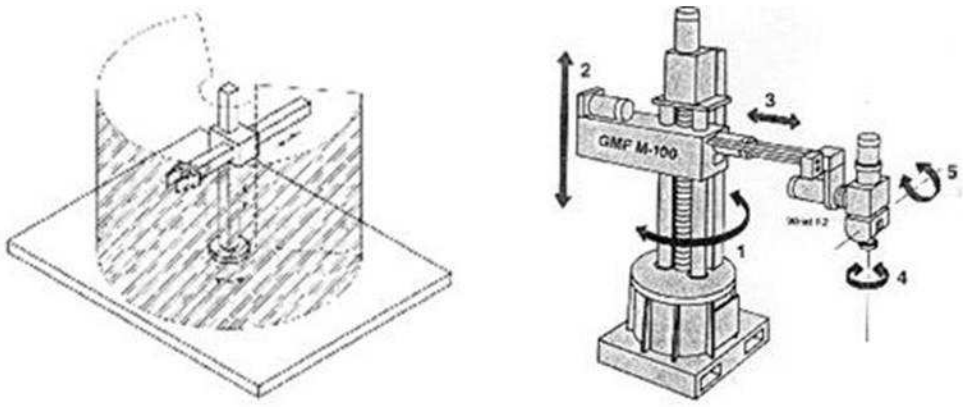

FIGURA 2 - MANIPULADOR DE GEOMETRIA CILÍNDRICA

A classe **Polar** ou **Esférica** (**RRP**) é caracterizada por as juntas da cintura e ombro serem rotacionais, sendo a restante prismática. O seu espaço de trabalho tem a forma de uma esfera, contudo nem todos os pontos da mesma estão acessíveis, sendo isso

mesmo visível na Figura 3. Este tipo de classe possui então as seguintes vantagens: localização da garra facilmente descrita com coordenadas polares; abrangência de um largo espaço de trabalho. No entanto, possui também as seguintes desvantagens: modelo cinemático complexo; difícil visualização dos limites do espaço de trabalho. À semelhança dos manipuladores da classe anterior, estes são usados em tarefas de transporte material a curtas distâncias, tendo a classe Polar a vantagem de poder mover material acima da altura da junta do ombro, uma vez que esta agora é rotacional, feito que na classe anterior não era possível.

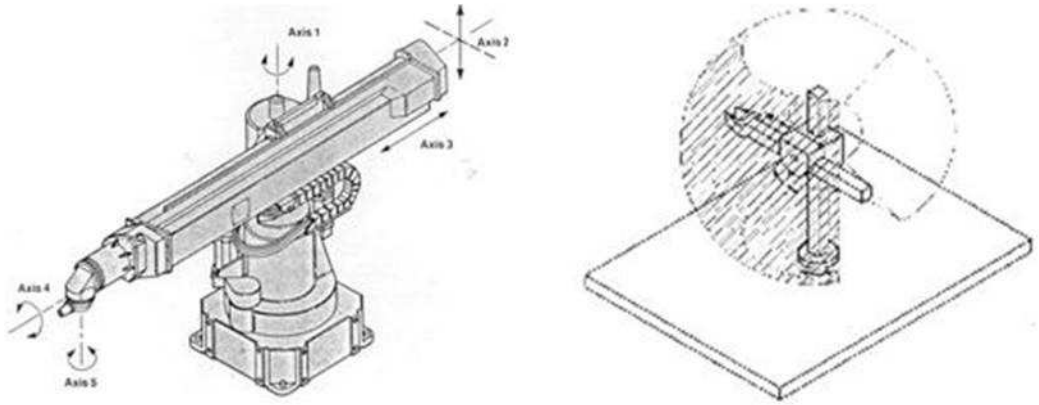

FIGURA 3 - MANIPULADOR DE GEOMETRIA POLAR OU ESFÉRICA

A classe seguinte é a **Articulada Vertical** ou também designada de **Antropomórfica** (**RRR**), onde todas as juntas, cintura, ombro e cotovelo, são rotacionais. Um exemplo desta geometria está na Figura 4, onde também é possível verificar que o seu espaço de trabalho é uma esfera, estando todos os pontos da mesma acessíveis. Tal como todas as outras classes, esta também possui vantagens e desvantagens que seguidamente serão enumeradas. Vantagens: máxima flexibilidade; largo espaço de trabalho comparativamente com o volume do manipulador; uso exclusivo de juntas rotacionais que facilmente podem impedir a entrada de pó e líquidos; acesso a objectos que estejam localizados sob outros em qualquer ponto do seu espaço de trabalho. Desvantagens: modelo cinemático complexo; difícil visualização dos limites do espaço de trabalho; controlo de movimentos lineares algo complexo. Apesar das desvantagens que este tipo de manipulador apresenta é o mais utilizado no mundo industrial, tendo especial enfoque em unidades industriais que possuem linhas de montagem onde, por exemplo existam tarefas de soldadura e pintura. No mercado internacional existem inúmeros fabricantes desta última classe de manipuladores, tais

como a ABB com o seu modelo IRB 1410, a Fanuc com o seu modelo ARC Mate 120iC ou a Kuka com o seu KR 1000 titan. As mesmas marcas possuem também outros modelos para as restantes geometrias anteriormente referidas.

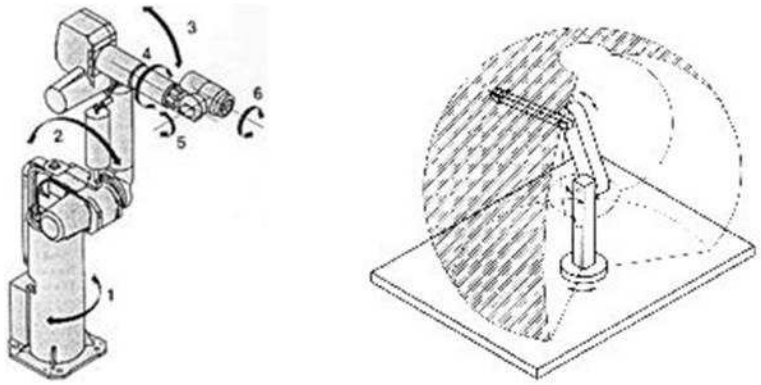

FIGURA 4 - MANIPULADOR DE GEOMETRIA ARTICULADA VERTICAL OU ANTROPOMÓRFICA

A última classe de manipuladores industriais tem o nome de **Articulada Horizontal** ou **SCARA** (**RRRP**). SCARA é acrónimo de *Selective Compliance Automatic Robot Arm*. Esta classe de manipuladores nasceu graças à necessidade que o mundo industrial sentiu em ter um braço de geometria simples. A Figura 5 apresenta a geometria desta classe. Nela pode-se observar que as três primeiras juntas são rotacionais e a última junta prismática. Com as três juntas rotacionais estes robots conseguem atingir qualquer ponto localizado no plano horizontal do seu espaço de trabalho. A junta prismática, onde se localiza o elemento terminal, tem como objectivo, dar profundidade ao espaço de trabalho do manipulador, fazendo com que este tenha a forma de um cilindro. Por último convém realçar que, embora este tipo de manipuladores tem um espaço de trabalho idêntico aos da classe Cilíndrica, estes apresentam geometrias distintas.

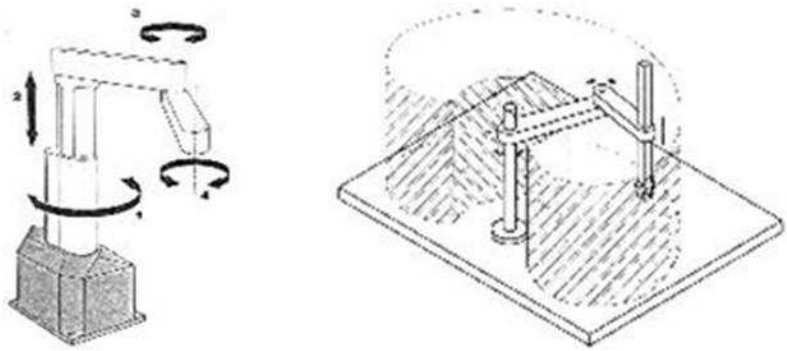

FIGURA 5 - MANIPULADOR DE GEOMETRIA ARTICULADA HORIZONTAL OU SCARA

Os manipuladores industriais são máquinas, como tal não são dotadas de qualquer tipo "inteligência". Para que estas façam aquilo para as quais foram programadas é necessário que a estas estejam associadas a um sistema de controlo. Tipicamente um manipulador, e todas as máquinas que envolvam movimento, têm associado o sistema de controlo, normalmente em malha fechada, presente na Figura 6. Por vezes são também utilizados sistemas de controlo de malha aberta, contudo em aplicações puramente didácticas onde são usados motores de passo. Cada bloco presente no diagrama de blocos da Figura 6 representa um elemento de um robot. De seguida serão apresentadas as principais características e funções de cada um destes blocos.

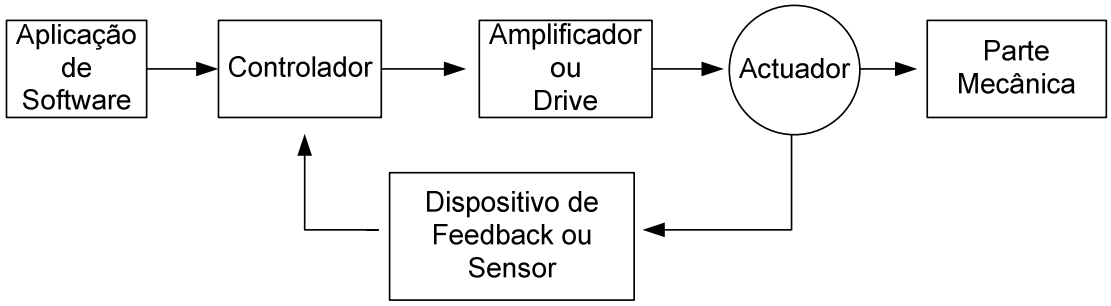

FIGURA 6 - DIAGRAMA DE BLOCOS DO SISTEMA DE CONTROLO DE UM MANIPULADOR INDUSTRIAL

O bloco de software tem por função fazer o interface com o utilizador. É este bloco o responsável por gerar os comandos a ser enviados para o controlador. Esses comandos podem ter múltiplos fins: podem ser comandos de configuração, comandos para definição da trajectória e muitos outros. Normalmente, este bloco está fisicamente localizado no controlador, fazendo-se o interface com o utilizador através de uma consola que pode pertencer, ou não, ao controlador. Para a construção desta aplicação existem inúmeras plataformas de software específicas nesta área, tendo praticamente cada fabricante de motores e respectivos *drives* o seu software próprio que permite fazer o interface e o respectivo planeamento de trajectória. Exemplos dessas plataformas de software são o RSLogix 5000 da *Rockwell Automation* que possui um pacote especial de nome *Kinematics*. Outro fabricante que possui software para realizar o planeamento e controlo de trajectória é a *Beckhoff* com a plataforma *TwinCAT* NC. Finalmente o software utilizado neste trabalho foi o NI-SoftMotion da *National Instruments*. Este foi o escolhido uma vez que a empresa onde foi realizado o trabalho descrito nesta dissertação já possuía todas as licenças para a sua utilização,

além de os seus responsáveis terem todo um *know-how* sobre o software da empresa *National Instrumens*. Convém referir que as empresas que fornecem hardware fornecem normalmente também software de ligação que permita a utilização deste material em aplicações realizadas com softwares de outros fabricantes. Isto acontece independentemente dos fabricantes de hardware terem, ou não, software próprio para realizar a aplicação pretendida. Desta maneira é possível criar uma aplicação onde hardware e software sejam feitos por empresas distintas, sendo esta prática bastante corrente no mundo empresarial.

O bloco seguinte ao software é o controlador. Este é o "cérebro" de todo o sistema de controlo. O controlador, tal como o nome indica, tem por missão controlar a desempenho dos actuadores. Este bloco recebe da aplicação a posição pretendida pelo utilizador, recebendo também a posição real do actuador através dos próprios *drives* dos motores (ou de sensores específicos para o efeito). Com estas duas entradas, e mediante o algoritmo de controlo utilizado, o controlador envia para o *drive* informação que permita a este saber o que devem os actuadores fazer. Desta maneira o controlador funciona como que um descodificador, que recebe da aplicação de software comandos de alto nível, entre outros a posição e velocidade pretendida, e mediante a posição real dos actuadores, determina a actuação que deve ocorrer no sistema. Até há pouco tempo a informação que era enviada para o *drive* era um sinal analógico, tipicamente entre -10 V e 10 V. Contudo começam a surgir *drives* capazes de a receberem informação na forma digital. Com esta inovação os dispositivos de IO existentes entre controlador e *drive*, tipicamente conhecidos por cartas de entrada e saída, deixam de ser necessários, sendo a informação transportada por protocolos de tempo real, como por exemplo *CANopen* [4], *EtherCAT* [5] ou *Profibus* [6]. Para garantir que as tarefas prioritárias do processador sejam atendidas no menor intervalo de tempo possível, são utilizados sistemas operativos de tempo real no controlador, fazendo com que o processador deixe de estar partilhado com outras aplicações. No mercado existem uma quantidade elevada de controladores acessíveis a praticamente todas as bolsas. A escolha de um determinado tipo de controlador depende muito da aplicação em que vai ser utilizado. A aplicação que se pretende desenvolver requer do controlador, elevado poder de cálculo e rapidez, como tal é necessário escolher um

controlador com características que o permitam fazer. Para o patamar de desempenho pretendido, no mercado existem entre muitos a série CX da empresa alemã *Beckhoff*, o Compact Non-Display Computer (200R) da empresa *Rockwell Automation*, o CompactRIO e o PXI da *National Instruments*. O controlador inicialmente escolhido foi o CampactRIO da *National Instruments* devido à facilidade de programação, baixo custo, elevado desempenho, e grande versatilidade, podendo suportar diversos módulos de entrada e saída, contudo a escolha final recaiu sobre o controlador PXI-8176, também ele da *National Instruments*, uma vez que já existia na empresa onde foram realizados os trabalhos. Esta escolha ficou a dever-se ao facto de este tipo de controlador poder suportar um módulo de comunicações compatível com o protocolo *EtherCAT*.

O bloco designado como amplificador ou *drive* no diagrama de blocos da Figura 6 não é mais do que unidade de potência dos actuadores. Este tipo de dispositivos recebe do controlador um sinal de comando que é amplificado e convertido para um valor de corrente eléctrica, corrente essa que vai ser injectada nos motores (actuadores), fazendo com que o movimento produzido seja proporcional ao sinal de comando enviado pelo controlador. Tipicamente, o sinal de controlo representa a velocidade de rotação desejada pelo controlador, contudo este tipo de dispositivos pode também aceitar sinais de controlo que representem a posição desejada ou mesmo o binário pretendido para os motores. [7] Os *drives* mais recentes oferecem ainda a possibilidade de eles próprios fecharem a malha de controlo, ou seja, têm a capacidade de receber e controlar a velocidade e/ou posição actual do motor. Se esta for diferente do especificado pelo sinal de controlo então o *drive* actua de forma imediata sobre o motor, fazendo com que este convirja para o valor especificado pelo controlador. Tal como já atrás foi referido, actualmente a troca de informação entre o controlador e os *drives* começa de ser feita de forma digital, recorrendo para tal a protocolos de comunicação de tempo real como por exemplo o *CANopen*, *Profibus* ou *EtherCAT*. Por fim falta apenas referir que existem mais tipos de *drives* para além dos eléctricos, como os *drives* hidráulicos e pneumáticos. O uso *drives* eléctricos e consequentemente motores eléctricos deve-se ao facto de estes terem alta precisão, fácil transporte e instalação, custo relativamente baixo e a carga a transportar ou

tratar ser por norma de pequenas ou médias dimensões. No mercado existem *drives* deste tipo para todas as aplicações, contudo apenas vão ser apresentados *drives* eléctricos para motores síncronos. Assim existe, empresa *SEW-EuroDrive* com o modelo MOVIDRIVE 61B, a empresa alemã *Siemens* com a família de *drives* SINAMICS e ainda a empresa *Beckhoff* com a família de *drives* AX2000 e AX5000. A escolha recaiu sobre a última solução apresentada, tendo contribuindo em grande medida o facto de este modelo possuir já um protocolo de comunicações com o controlador, no caso o *EtherCAT*.

O bloco da Figura 6 designado de actuador recebe um sinal em corrente emitido pelo respectivo *drive* ou amplificador. A acção dos actuadores é proporcional ao sinal de controlo emitido pelo controlador graças à conversão que é feita pelo *drive* de cada actuador. Existem três tipos de actuadores: eléctricos, hidráulicos e ainda pneumáticos. A tabela 1 apresenta uma pequena síntese das vantagens e desvantagens do uso de cada um destes tipos de actuadores. No parágrafo anterior foi referido que na área da robótica de manipuladores, quer *drives*, quer actuadores seriam eléctricos, sendo estes últimos motores DC sem escovas (*Brushless*) de ímanes permanentes. Para a escolha deste tipo de motores pesou o facto de estes possuírem elevada fiabilidade e uma reduzida necessidade de manutenção, tendo contudo um preço mais elevado que motores com escovas. Na construção de uma aplicação do tipo da que se pretende realizar, o primeiro passo a ser feito é, justamente proceder ao acerto dos parâmetros do compensador do motor, fazer o chamado *tuning* do compensador. O compensador de um motor, tem por missão fazer adaptar a resposta do motor às características pretendidas. Existem vários tipos de compensadores sendo os principais para o controlo de dispositivos mecânicos simples como motores, o *on-off*, o proporcional, o proporcional diferenciador usualmente designado por PD, e ainda o proporcional integrador derivativo, tipicamente conhecido por PID. A escolha de um destes compensadores deve ser feita tendo em conta as características que se pretendem para a resposta do motor. Na área da robótica de manipuladores é prática corrente o uso do controlador proporcional integrador derivativo ou PID. O ajuste dos parâmetros deste compensador é feito através de software, ou hardware, disponibilizado pelas empresas fabricantes deste tipo de componentes. Para esta

dissertação, a família de motores escolhidos foi a AM30XX da empresa *Beckhoff*. Poder-se-ia apresentar mais motores, contudo, e como já atrás foi referido, a escolha de um motor está intimamente ligada com a escolha do respectivo *drive*, para que o acerto dos parâmetros do compensador do motor seja feito de maneira simples e eficaz.

TABELA 1 - COMPARAÇÃO DOS PRINCIPAIS TIPOS DE ACTUADORES

Características	Actuadores		
	Eléctrico	Hidráulico	Pneumático
Controlo	Fácil integração em aplicações de controlo complexo	Boa integração quando são utilizadas electro servo-válvulas	Complexo devido ao controlo da compressibilidade do ar
Velocidade	Grandes	Moderada	Elevadas
Precisão	Boa, contudo limitada pelo uso de transmissão	Boa	Má,
Custos	Relativamente baixo, e com reduzida manutenção	Elevados, agravados pela necessidade de manutenção permanente	Relativamente baixos
Dimensão e Peso	Reduzidos	Elevados	Elevados

O único bloco da Figura 6 que ainda não foi apresentado é o de dispositivos de feedback. Este bloco faz com que a malha de controlo se feche, medindo directamente no actuador uma grandeza que é transmitida para o controlador ou para o *drive*, mediante o sistema de controlo implementado. A, ou as, grandezas medidas pelos dispositivos de feedback são tipicamente três: a velocidade, a posição do eixo do motor ou o binário. Embora na Figura 6 este bloco apareça como sendo um bloco independente, é regra geral todos os motores terem já no seu interior este tipo de dispositivos de medida (sensores). Os dispositivos mais usados para a medição de posição e velocidade são os *encoders* ópticos e também os *resolvers*. Por vezes para a medição a velocidade são também usados tacómetros. Como estes dispositivos normalmente já se encontram integrados no interior dos motores, não será apresentado nenhum exemplo deste tipo de dispositivos existentes no mercado.

Como já foi dito, o controlador é como que o "cérebro" do sistema de controlo, podendo este ter vários tipos de controlo. Os algoritmos de controlo mais usados são quatro: o controlo de velocidade, o controlo de posição (ponto a ponto), o controlo de binário e ainda o chamado controlo e*lectronic gearing* (método onde o movimento de um motor é sempre referente a um segundo motor que serve de referência para todos os outros). [8] A implementação de um destes tipos de algoritmos de controlo depende essencialmente das especificações do processo a controlar, dependendo também do tipo de dispositivos de feedback disponíveis. Existem muitas maneiras de se controlar a trajectória de um movimento. Para os primeiros dois tipos de controlo, velocidade e posição, são usados perfis de movimentos. Nos perfis de movimento é definida a variação da velocidade, ou da aceleração, de um motor de um eixo ao longo do tempo. Os perfis mais usados são o trapezóide, o perfil de curva de curva em S e ainda o perfil de triângulo, podendo haver perfis personalizados pelo programador. A Figura 7 ilustra o perfil trapezóide e também o perfil curva em S. Para o controlo do binário do motor é necessário que este tenha um dispositivo que consiga efectuar a medição do binário aplicado, e enviar essa mesma informação para o controlador através da malha de feedback. Quanto ao quarto tipo de controlo, e*lectronic gearing*, este é usado para sincronizar o movimento de um ou mais eixos *slaves*, com um dispositivo *master*. O dispositivo *master* pode ser um *encoder* de um outro eixo, uma ADC, ou mesmo um gerador de trajectória de outro eixo. Este algoritmo faz com que o *slave* acompanhe o movimento do *master* numa determinada proporção - *gear ratio*. Se, por exemplo, se definir que o *gear ratio* de um determinado *slave* é igual a dois, então esse eixo efectuará duas rotações por cada rotação do seu *master*. Este algoritmo pode também ser implementado mecanicamente recorrendo a engrenagens, contudo é prática geral a implementação electrónica devido a versatilidade que este tipo de implementação oferece.

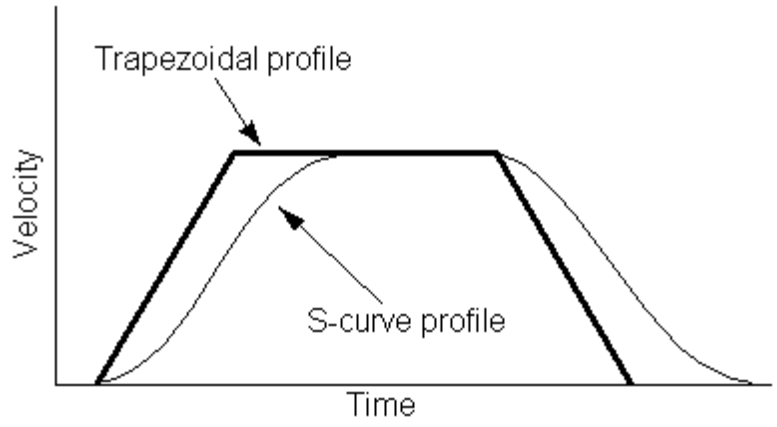

FIGURA 7 - VARIAÇÃO DA VELOCIDADE COM O TEMPO PERFIL TRAPEZÓIDE E CURVA EM S

Depois de explicado um pouco de todo o funcionamento do sistema de controlo de um manipulador deste tipo, vão agora ser apresentadas as duas técnicas mais usuais de programação destes dispositivos. Um braço robótico antropomórfico pode ser programado através de trajectórias pré definidas pelo programador, ou "ensinado" para a sua função. Na primeira técnica o programador tem que conhecer ante mão toda a trajectória que o robot deve efectuar, e tem também que conhecer o tipo de movimento pretendido. O elemento terminal pode ter três tipos de movimento: em linha recta, em arco ou ainda ambos. O outro método de programação, ensino, é bem mais fácil e intuitivo do que o primeiro método. O manipulador é "ensinado" pelos seus operadores a executar uma determinada tarefa, repetindo-a de seguida as vezes que for necessário. Este método é tipicamente usado para programação de manipuladores que executem tarefas rotineiras de menor complexidade.

O braço mecânico que se pretende controlar, tal como já foi dito, possui seis graus de liberdade, como tal possui também seis juntas rotacionais que permitam fazer o controlo da trajectória e também a localização do elemento terminal. Também já foi dito que a posição num determinado momento do manipulador é definida pela junta da base e as duas seguintes, estando a orientação do elemento terminal a cargo das três ultima juntas. Falta agora saber em que medida actuar individualmente em cada junta, para que o elemento terminal atinja a posição pretendida. A esse estudo dá-se o nome de cinemática de um manipulador. A cinemática de um manipulador é então o estudo do conjunto de relações entre as posições, velocidades e acelerações dos seus

elos. [3] Para descrever a cinemática de um manipulador existem duas possibilidades: a cinemática directa e a cinemática inversa, sendo que a cada uma destas está associada uma problemática diferente. A cinemática directa está associada à problemática do cálculo da posição/orientação do elemento terminal, a partir da posição angular de cada junta. A cinemática inversa de um manipulador trata da problemática inversa da cinemática directa, ou seja, calcula a posição angular de cada junta, partindo da posição/orientação do elemento terminal. O diagrama da Figura 8 ilustra o conceito de cinemática directa e inversa. Logo no início deste capítulo foi referido que os robots manipuladores industriais de seis eixos possuíam algumas posições singulares, contudo, e à semelhança do braço humano, são dispositivos redundantes. Este facto leva a que a, fora das singularidades, a cinemática inversa apresente para a mesma posição/orientação do elemento terminal, uma ou mais soluções para a posição angular de cada junta.

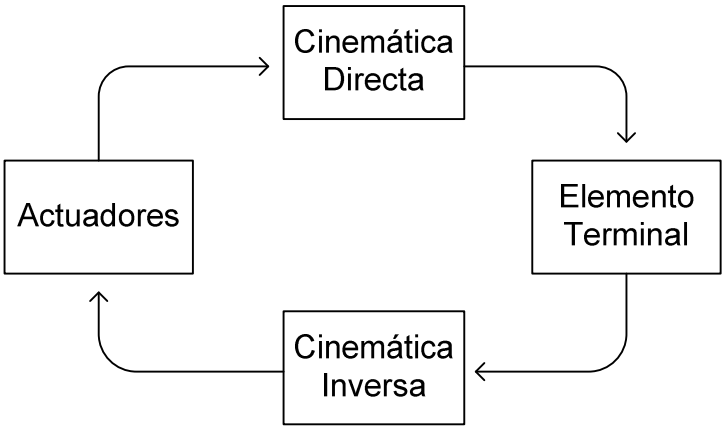

FIGURA 8 - CINEMÁTICA DIRECTA E INVERSA DE UM MANIPULADOR

O parágrafo anterior descreve as relações entre coordenadas do elemento terminal do manipulador e a posição angular dos vários actuadores. A cinemática directa e a cinemática inversa não dão contudo qualquer informação sobre as características do movimento entre duas quaisquer posições. Dito de uma outra maneira, os dois anteriores tipos de cinemática não dizem qual o incremento, ou decremento que é necessário fazer nas juntas para que o elemento terminal atinja uma determinada posição com um determinado tempo. Essa tarefa fica a cargo da chamada cinemática diferencial. Este tipo de trabalho é por norma feito pela aplicação informática que, no

caso desta dissertação, já se encontra implementada no correspondente pacote de software existente.

Para que o estudo sobre os movimentos do manipulador esteja completo falta apenas abordar a dinâmica. A dinâmica de um manipulador trata do problema de relacionar as forças exercidas no robot com o seu movimento, isto é, com as velocidades e acelerações das juntas.

Para a programação de um manipulador é necessário saber como se relacionam as diversas juntas (i.e., motores) com a posição espacial do elemento terminal. Tal tarefa está a cargo da cinemática directa ou inversa. É também necessário saber a relação que existe entre as juntas e a dinâmica que estas devem ter ao longo do tempo para que o seu elemento terminal atinja uma determinada localização, tal tarefa fica a cargo da cinemática diferencial. Por fim, é necessário efectuar o chamado planeamento da trajectória. É nesta fase que se estudam métodos que permitem definir os regimes de velocidade dos diversos actuadores de forma a fazer o manipulador cumprir os objectivos de movimentação ou deslocamento planeados. Para o estudo da variação da velocidade já foram anteriormente apresentados os algoritmos mais utilizados na definição de perfis. Um robot industrial pode realizar tarefas que envolvam a simples movimentação de um ponto para outro do elemento terminal, num espaço de tempo pré-definido, tarefas de paletização ou transferência, ou também pode realizar tarefas de alguma complexidade que envolvam a execução de trajectórias bem definidas no espaço operacional em intervalos de tempo bem determinados. Exemplos dessas tarefas são os casos da soldadura e pintura. Para a realização do primeiro exemplo de tarefas usam-se movimentos ponto-a-ponto enquanto para o segundo usam-se movimentos de trajectória contínua. No primeiro tipo de trajectória o percurso efectuado pelo manipulador entre o ponto inicial e final não é conhecido, enquanto que no segundo tipo de abordagem todo o percurso é definido *a priori* de forma analítica ou numérica. Existe ainda o meio termo entre estas duas abordagens que é a definição de pontos intermédios de passagem ao longo do percurso entre o ponto inicial e final do elemento terminal. Esta última abordagem é a mais versátil e a mais usada na prática. Na aplicação desenvolvida irá existir uma função que actua ciclicamente com um período de alguns milissegundos, onde se

geram pontos de passagem para o manipulador. Desta forma a trajectória é continua sendo constituída por pontos de passagem muito próximos uns dos outros.

Ao longo dos últimos anos a robótica tem tido um desenvolvimento espantoso, mas é contudo ainda uma jovem ciência que tem algumas arestas para limar. Ao nível prático existe o problema de comunicações entre dispositivos que não sejam do mesmo fabricante, uma vez que não existe um protocolo de comunicações standard para todas as marcas. Este problema é minorado com os servidores que são fornecidos pelos fabricantes de hardware, permitindo assim que o seu material seja incluído em aplicações de outros fabricantes. Um exemplo dessa prática é a empresa *Beckhoff* que utiliza o protocolo *ADS* (*Automation Device Specification*) para comunicar com todos os dispositivos por ela fabricados. A *Beckhoff* disponibiliza um servidor deste protocolo (*TwinCAT* IO OPC Server), além de todo o software de desenvolvimento próprio da empresa. Outro desafio que é colocado a esta jovem indústria é a criação de manipuladores de mais baixo custo, onde o retorno económico do investimento feito seja mais imediato. Essa questão é ainda mais importante se se tiverem em linha de conta que o tecido empresarial nacional e europeu é constituído essencialmente por pequenas ou médias empresas, para as quais um investimento num equipamento destes é algo significativo. Também a simplificação dos processos de programação deste tipo de equipamentos deve ser simplificado, pois, mesmo nos dias de hoje, as empresas que fornecem manipuladores robóticos necessitam de técnicos altamente qualificados. Por último, mas não menos importante, a robótica tem que resolver o problema social e moral que o seu desenvolvimento pode provocar. Até que ponto se deve desenvolver uma máquina que possa substituir por completo o homem nas suas tarefas? Todas essas questões devem ser respondidas por esta formidável ciência.

Depois de se ter explicado um pouco dos conceitos base para a realização desta dissertação, vai agora ser apresentada a solução que se pretende implementar. Será usado o controlador PXI-8176 (*National Instruments*) para se efectuar o planeamento e controlo de trajectórias do robot manipulador. O controlador terá um sistema operativo de tempo real, onde estará a correr uma aplicação desenvolvida pelo pacote de software *NI-SoftMotion* da linguagem de programação gráfica *Labview*. A aplicação que se pretende desenvolver será constituída por três blocos: interface, comunicação

e actuação. Dentro deste último bloco será efectuada toda a tarefa de cálculo de trajectória e controlo. Será necessário efectuar o estudo da cinemática directa e inversa do manipulador para que se possa relacionar a posição angular dos motores, com a posição real do elemento terminal. Os actuadores de cada junta serão motores eléctricos síncronos DC sem escovas, da família de motores AM3000, da empresa *Beckhoff*. Para polarizar, controlar e configurar estes motores serão usados *drives* da família AX50000, também da empresa *Beckhoff*. A comunicação entre *drive* e controlador é crítica, como tal será usado um protocolo de comunicações de tempo real - o protocolo *EtherCAT*. Como o controlador escolhido ainda não possui qualquer porta de comunicações para este protocolo, será então necessário introduzir o módulo de comunicação *NI* PXI-8231 (*National Instruments*). Todos os dados de feedback serão obtidos por dispositivos internos do motor, sendo estes posteriormente enviados para o controlador através do seu respectivo *drive*. Toda a interface com o utilizador ficará a cargo de uma consola externa ligada em rede ao controlador. Através dessa interface o utilizador pode observar toda a informação relativa ao funcionamento do braço robótico, e se se pretender, programá-lo para a realização de uma tarefa através do modo de ensino.

2.2.Enquadramento

A ciência da robótica trata de máquinas multifuncionais programáveis que podem ser afectas a tarefas normalmente desempenhadas por seres humanos. Estas máquinas têm a vantagem de ser completamente versáteis, podendo, por exemplo, ser afectadas a uma determinada tarefa e, mediante o meio e as condições existentes, adaptar-se a esta sem grandes dificuldades. Os seres humanos não possuem tal versatilidade, sendo na maioria resistentes a qualquer tipo de alteração do seu processo produtivo.

Os manipuladores industriais conheceram ao longo destes últimos anos desenvolvimentos significativos. Toda esta evolução veio dotar o mundo industrial de uma característica que até aqui era desconhecida - a flexibilidade. Hoje em dia, a riqueza de um País é medida pelo seu produto interno bruto (PIB), e para este

contribui em grande medida o grau de desenvolvimento e automação do seu processo produtivo. Uma economia que seja baseada em industrias com processos produtivos flexíveis tem tendência a ser muito mais produtiva que outra baseada em processos rígidos de produção. Nas décadas de 50, 60, 70 e 80, toda a indústria mundial era baseada na chamada automação rígida, onde uma máquina era adstrita a uma determinada tarefa, e jamais no seu tempo de vida útil realizava outra. Com este tipo de processo produtivo existia, por parte da indústria, uma grande oposição à inovação e desenvolvimento de produtos que implicassem a alteração da sua linha de montagem. Estas quatro décadas ficaram conhecidas como os anos dourados da produção industrial.

Com a queda do muro de Berlim, e a consequente globalização mundial, deixaram de existir fronteiras comerciais. Os produtos têm hoje tempos de vida cada vez mais limitados. Aliada às restrições temporais destes novos produtos, surge a necessidade destes terem cada vez mais qualidade, e menor custo. Perante este novo paradigma mundial as empresas deixaram de produzir modelos em larga escala, passando a produzir em pequena ou média escala. Para acompanhar esta nova exigência de mercado, a indústria teve que substituir todo o seu processo produtivo rígido e obsoleto, por um novo que fosse flexível adaptado às novas exigências de mercado.

Actualmente, a sociedade mundial é uma sociedade consumista que tem um crescente poder de compra. Em países como a Índia e China, que tradicionalmente tinham, e continuam a ter nos dias de hoje, grandes assimetrias sociais, têm hoje uma sociedade altamente consumista onde as classes de estrato social mais baixo têm hoje um crescente poder de compra. A indústria também tem contribuído em grande medida para o consumismo que hoje existe nessa e em todas as sociedades, introduzindo nas pessoas a necessidade de estas adquirem determinado produto a fim de serem reconhecidas em certos meios sociais. Estas novas características de mercado favorecem a denominada “Zona da Robótica”(ver gráfico 1) sendo esse o principal motivo para que hoje em dia se assista a uma crescente “robotização” do mundo.

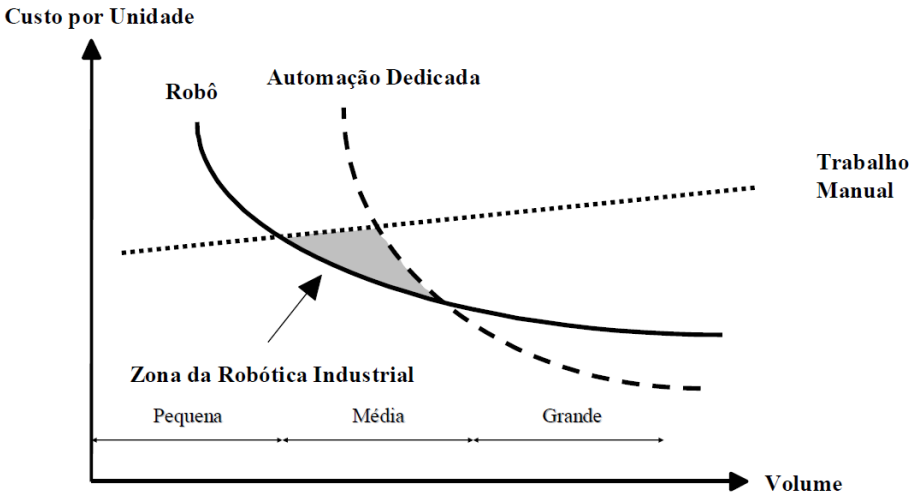

GRÁFICO 1 - ZONA DA ROBÓTICA INDUSTRIAL [1]

Como hoje em dia é sabido, as empresas recorrem cada vez em maior número à robótica para conseguir que os seus produtos tenham uma maior qualidade e mais baixo custo, contudo existem ainda outras motivações para o seu uso. Com este tipo de máquinas as empresas passaram a disponibilizar de uma capacidade produtiva de bens muito mais diversificada. Outra das motivações é o curto espaço de vida dos produtos que essa mesma indústria produz. Esta última motivação reveste-se de especial importância se se tiver em conta a sociedade capitalista em que vivemos.

Este tipo de motivações levou à adopção de um sistema de produção que seja altamente flexível. Neste conceito produtivo os robots manipuladores industriais são especialmente importantes pois são máquinas programáveis que permitem um planeamento de todas as suas tarefas e uma consequente execução de forma precisa, repetitiva e relativamente barata. Convém realçar o facto de estas máquinas possuírem nos dias de hoje uma forte componente de comunicações de entrada - saída que torna possível uma fácil integração em sistemas de produção já existentes.

Os robots industriais são actualmente máquinas altamente sofisticadas, como tal acarretam custos que são significativos aquando da compra e manutenção ao longo da vida útil da máquina. Este tipo de dispositivos, embora tendo uma maior eficiência em certas tarefas, necessitam de engenheiros altamente qualificados no seu desenvolvimento, de operários capazes de as programar, utilizar e de efectuar a sua manutenção diária. Para instalar, operar e manter todos os dispositivos existentes

numa linha de produção industrial, são necessários técnicos cada vez mais qualificados, pois o grau de complexidade destas verdadeiras redes informáticas tem aumentado gradualmente com o tempo.

Comparando as motivações com os custos, observa-se que esta indústria ainda tem que criar soluções para problemas relacionados com o projecto e desenvolvimento destes dispositivos sendo este um desafio aliciante, motivador do interesse de engenheiros por este tipo de tarefa. É, assim, necessário criar aplicações informáticas mais acessíveis que permitam a sua programação e consequente simulação, sendo também necessário formar mais técnicos e engenheiros capazes para trabalhar com este tipo de equipamento.

Capítulo 3

3. Conceitos teóricos

3.1.Localização de um objecto

A posição de um objecto, qualquer que seja, pode ser descrita no espaço, ou no plano, através de sistemas de eixos de coordenadas. Quando um objecto se encontra localizado no plano usa-se um sistema de coordenadas com duas dimensões, contudo quando o mesmo objecto de encontra localizado no espaço usa-se um sistema de eixos de coordenadas com três dimensões.

Na robótica, e nesta dissertação, os sistemas de eixos de coordenadas terão por base a regra da mão direita, ou seja o sentido positivo de rotação será definido pela regra da mão direita, como tal os sistemas de eixos serão directos, ou seja o sentido positivo de rotação é o anti-horário. A Figura 9 ilustra os sistemas directos de coordenadas a duas e a três dimensões.

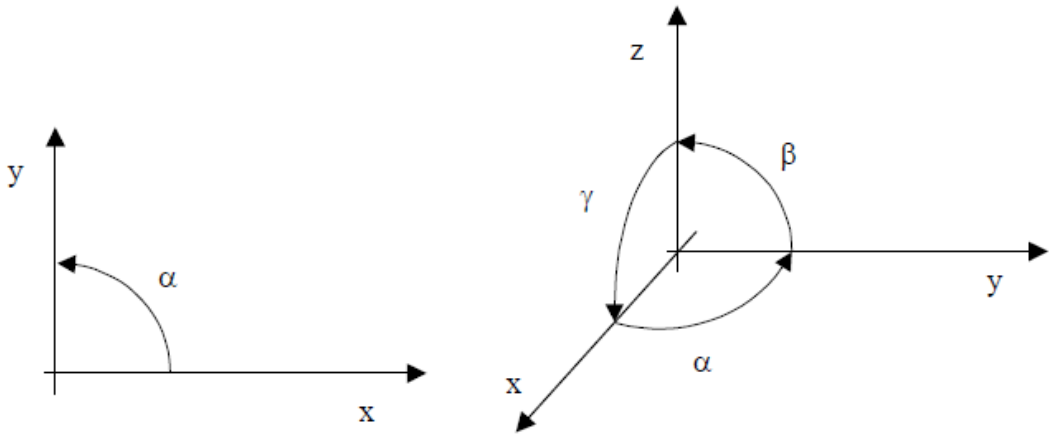

FIGURA 9 - SISTEMAS DIRECTOS DE COORDENADAS A NO PLANO E NO ESPAÇO

Um ponto no espaço é representado tal como demonstra a Figura 10 por um vector de três coordenadas, (no caso do plano é apenas por duas coordenadas). O

termo vector surge por norma associado ao conceito de movimento numa determinada direcção e sentido. Quando é utilizado o termo de vector associado ao conceito de movimento ou de deslocação está-se a pensar num vector que é aplicado na origem do referencial e com as coordenadas do seu ponto extremo. Assim no caso da representação da caixa da Figura 10 dois dos vectores $\vec{p}$ e $\vec{q}$ têm as mesmas coordenadas que os pontos A e B respectivamente, contudo estão em causa dois conceitos formalmente distintos. [3]

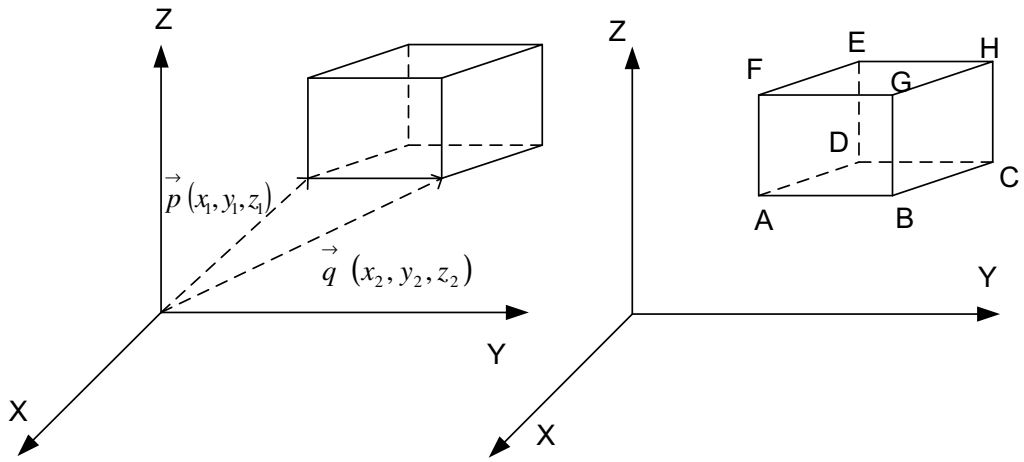

FIGURA 10 - LOCALIZAÇÃO DE UM OBJECTO NUM SISTEMA DE EIXOS TRIDIMENSIONAL

Considere-se agora que a caixa da Figura 10 se move no sentido positivo do eixo YY. Esta translação implica o aparecimento de oito novos vectores de definição da caixa. O cálculo dos oito novos vectores de localização é moroso, como tal este não é um processo atractivo uma vez que quando ocorrem transformações mais complexas do que uma simples translação o cálculo seria extremamente complexo. A solução passa pela definição de um novo sistema de eixos, no objecto, essa mesma solução está presente na Figura 11. Desta maneira, sempre que a caixa é movida, é criado um novo referencial junto ao objecto para que o objecto seja descrito segundo este, novo referencial, no caso designa-se por N. O vector que une a origem do referencial R com a origem do novo referencial, N, é calculado através da expressão presente na Equação 1, onde $\vec{p}$ representa o vector que une a origem do referencial R com a origem do novo referencial, N.

EQUAÇÃO 1

$$\vec{r} = \vec{p} + \left(\vec{r} - \vec{p} \right)$$

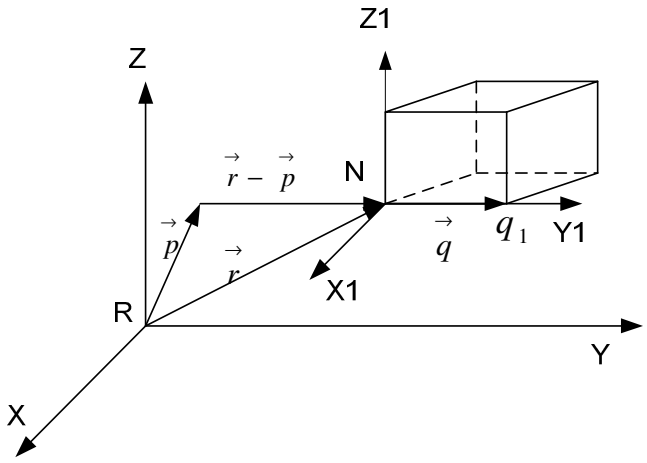

FIGURA 11 - LOCALIZAÇÃO DA CAIXA SEGUNDO O REFERENCIAL ASSOCIADO AO OBJECTO

Depois de explicado o processo de localização do novo referencial de referência do objecto vai agora proceder-se à explicação de como obter os novos pontos da caixa segundo o referencial inicial. Qualquer ponto no espaço pode ser visto, ou designado, de diferentes formas consoante o referencial usado. Fisicamente é sempre o mesmo ponto, a sua descrição é que muda dependendo do referencial associado a este. [3] A nova localização do ponto segundo o referencial original pode ser calculada através da expressão Equação 2, onde: ${}^{R}r$ representa a localização do vector r segundo o referência R, ${}^{R}q$ representa a localização do vector q segundo o referencial R e ${}^{R}q_1$ representa a localização do ponto q_1 segundo o referencial R. Assim o problema de calcular a nova localização de um objecto é reduzido ao cálculo da relação entre dois referenciais. Depois de calculada a relação entre os dois referenciais é possível calcular a nova localização do objecto segundo o referencial original. [2]

$${}^{R}q_1 = {}^{R}r + {}^{R}q$$ **EQUAÇÃO 2**

No exemplo usado anteriormente efectuou-se uma translação simples no eixo yy, contudo, tudo se torna mais complexo quando existem sucessivas translações intercaladas por rotações em um ou mais eixos. Um ponto representado no espaço tridimensional pode sofrer seis transformações geométricas, três translações e três rotações elementares. As translações podem ocorrer segundo o eixo xx, $Trans(x,a)$, segundo o eixo yy, $Trans(y,a)$ ou então segundo o eixo zz, $Trans(z,a)$em que *a* é o

valor do deslocamento. No que toca às rotações podem ocorrem também uma em cada eixo, rotação em torno do eixo xx $Rot(x,\theta)$, em torno do eixo yy $Rot(y,\theta)$ ou em torno do eixo zz $Rot(z,\theta)$ sendo θ o valor do ângulo de rotação. Todas as transformações geométricas podem ser visíveis a Figura 12. De forma natural se deduz que qualquer transformação geométrica pode ser decomposta em transformações elementares, translações e/ou rotações. [2]

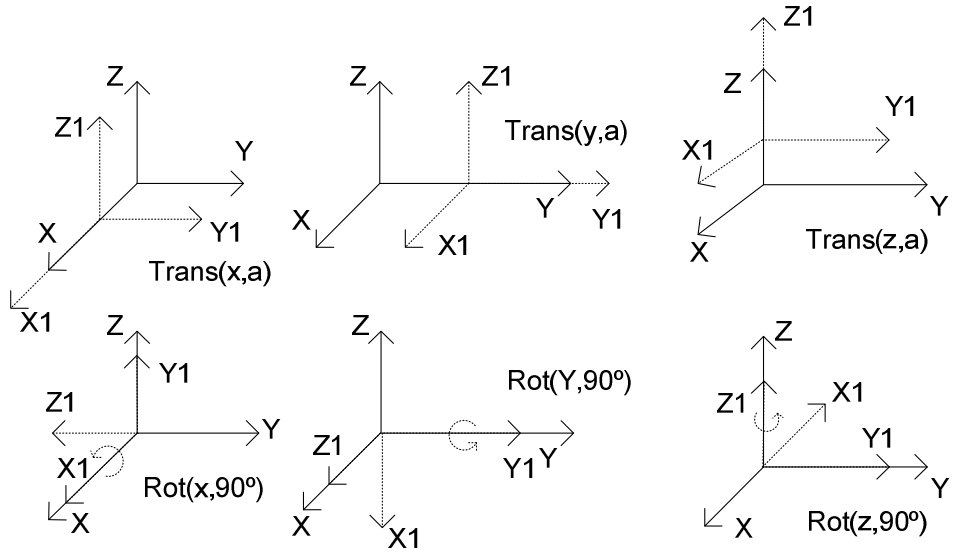

FIGURA 12- SEIS TRANSFORMAÇÕES GEOMÉTRICAS ELEMENTARES

3.2.Matriz de transformação

Tal como já foi dito no ponto anterior, a nova localização de um objecto pode ser feita recorrendo a um novo referencial que em relação ao referencial inicial sofreu uma transformação geométrica. Se o novo sistema de eixos tiver três dimensões este pode ser descrito por uma matriz constituída por quatro vectores, se o novo sistema de eixos tiver apenas duas dimensões a matriz que o descreve irá ter simplesmente três vectores. A matriz que descreve o novo sistema de eixos tem o nome de matriz de transformação e possui um vector que representa a translação feita pelo novo referencial, enquanto os restantes vectores representam a orientação dos novos eixos do sistema de coordenadas. A matriz de transformação irá representar todas as rotações e translações que o novo referencial sofreu em relação ao referencial anterior. Desta maneira, a multiplicação das sucessivas matrizes de transformação pode então descrever uma sequência de movimentos de um objecto.

3.2.1. Matriz de transformação no plano

Um ponto no plano pode ser descrito por duas coordenadas x e y. Considere-se agora que esse ponto sofre uma translação orientada segundo o vector $p = \lfloor p_x ; p_y \rfloor$, tem-se então o seguinte sistema de equações que define a nova localização:

$$\begin{cases} x_1 = x + p_x \\ y_1 = y + p_y \end{cases}$$ **EQUAÇÃO 3**

O mesmo sistema de equações sob a forma matricial tem a seguinte forma:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$ **EQUAÇÃO 4**

Se se considerar uma transformação geométrica, linear, absolutamente genérica onde cada nova coordenada passa a depender não só de um parâmetro independente, mas também de todas as coordenadas originais, obtêm-se o seguinte sistema de equações:

$$\begin{cases} x_1 = ax + by + p_x \\ y_1 = cx + dy + p_y \end{cases}$$ **EQUAÇÃO 5**

O mesmo sistema de equações representado na forma matricial tem a seguinte notação:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$ **EQUAÇÃO 6**

A matriz quadrada que surge na expressão anterior é designada por T, enquanto que o vector p se designa por termo independente. A matriz de constantes T pode ser usada para descrever transformações geométricas, isso mesmo é possível variando os valores das suas constantes. Se se considerar que: $p = \vec{0}$; $a = d = 1$; $b = c = 0$ e substituindo esses valores na equação anterior Equação 6, fica-se então com:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$ **EQUAÇÃO 7**

Como é possível verificar, quando T assume a forma de matriz identidade, desta maneira a matriz de transformação não traduz qualquer translação ou rotação, contudo se: $p = \vec{0}$; $d = 1; c = b = 0$:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} ax \\ y \end{bmatrix}$$ **EQUAÇÃO 8**

Na expressão anterior observa-se que a é um factor de escala na direcção do eixo xx. É de notar que o factor de escala e translação são duas coisas completamente diferentes. Em quanto que na translação existe uma deslocação do referencial segundo um determinado vector, no factor de escala uma, ou mesmo as duas coordenadas apenas vêm multiplicadas de uma constante. De forma semelhante se pode produzir um factor de escala segundo o eixo yy, bastando para tal que: $p = \vec{0}$; $d \neq 1; c = b = 0; a = 1$.

A matriz de transformação T também pode também ser usada para traduzir rotações. Isso mesmo é conseguido quando por exemplo:

$$T = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Quando a matriz T assume estes valores é efectuada uma rotação de 90^0 em torno do eixo ortogonal ao sistema de coordenadas, no sentido contrário ao dos ponteiros do relógio. Isso mesmo é traduzido na Equação 9:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} -y \\ x \end{bmatrix}$$ **EQUAÇÃO 9**

A matriz apresentada em baixo traduz a mesma rotação que a matriz T apresentada na Equação 9 quando $\theta = 90^0$. Desta maneira, pode-se então dizer que a matriz da Equação 10 é a matriz de rotação genérica, uma vez que esta pode traduzir

qualquer rotação que ocorra em torno do eixo ortogonal ao referencial, ou seja, em torno da origem do referencial.

$$T = \begin{bmatrix} \cos(\theta) & -sen(\theta) \\ sen(\theta) & \cos(\theta) \end{bmatrix}$$ **EQUAÇÃO 10**

Para ilustrar isso mesmo pode-se recorrer ao exemplo de um elo que roda em torno da origem do sistema de coordenadas. Na Figura 13 está presente o elo que inicialmente é rodado de ϕ e de seguida rodado de θ. A partir da referida figura é possível inferir o seguinte sistema de equações:

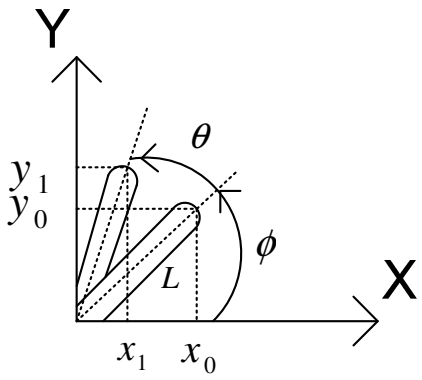

FIGURA 13 - ROTAÇÃO GEOMÉTRICA DE UM ELO NO PLANO

$$\begin{cases} x_1 = L.\cos(\theta+\phi) \\ y_1 = L.sen(\theta+\phi) \end{cases} \Leftrightarrow \begin{cases} x_1 = L.[\cos(\theta).\cos(\phi) - sen(\theta).sen(\phi)] \\ y_1 = L.[sen(\phi).\cos(\theta) + \cos(\phi).sen(\theta)] \end{cases}$$ **EQUAÇÃO 11**

Sabendo que:

$$\begin{cases} x_0 = L.\cos(\phi) \\ y_0 = L.sen(\phi) \end{cases}$$ **EQUAÇÃO 12**

Substituindo a Equação 12 na Equação 11 obtém-se:

$$\begin{cases} x_1 = x_0.\cos(\theta) - y_0.sen(\theta) \\ y_1 = y_0.\cos(\theta) + x_0.sen(\theta) \end{cases}$$ **EQUAÇÃO 13**

Finalmente a Equação 13 sob forma matricial virá:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -sen(\theta) \\ sen(\theta) & \cos(\theta) \end{bmatrix} . \begin{bmatrix} x_o \\ y_0 \end{bmatrix}$$ **EQUAÇÃO 14**

Pode-se então constatar que a matriz de transformação T, Equação 14, é equivalente à matriz de rotação genérica que anteriormente já tinha sido apresentada, na Equação 10. [3] Esta matriz quadrada de dimensão dois irá representar a rotação efectuada pelo novo referencial em relação ao original.

Como foi dito no início deste capítulo, a matriz de transformação no plano é constituída por três vectores. Destes três vectores, um representa a translação efectuada pelo novo referencial e os restantes representam a nova orientação do novo referencial. A sub matriz de rotação da matriz de transformação já foi apresentada na Equação 10, faltando agora apresentar o vector de translação da matriz de transformação. Com a introdução do vector de translação a matriz de transformação é então apresentada na Equação 15.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & p_x \\ c & d & p_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Equação 15

Esta nova matriz de transformação introduz assim o sistema de coordenadas homogéneas. É de notar que se se adicionar apenas o vector responsável pela translação na matriz de transformação da Equação 15, esta deixaria de ser uma matriz quadrada para passar a ser uma matriz de duas linhas e três colunas, sendo que se tal acontece-se esta deixaria de ter inversa, e também não se poderia efectuar a multiplicação da referida matriz pelo vector de localização do ponto inicial $a_0 = [x, y]^T$. Este aspecto é extremamente importante na robótica uma vez a nível computacional o cálculo seria muito mais complexo.

Para que tal não aconteça foi introduzida uma terceira coordenada, em ambos os vectores de localização dos pontos inicial, $a_0 = [x, y, 1]^T$, e final, $a_1 = [x_1, y_1, 1]^T$, com o valor constante de um. Também foi introduzida uma linha com dois zeros nas primeiras duas colunas da matriz de transformação e um um na terceira coluna. Este

último elemento da matriz assume um valor constante de um devido ao facto de ser um factor de escala unitário do vector de translação.

Tal como anteriormente já tinha sido avançado, a combinação dos parâmetros da matriz de transformação T permite descrever várias situações. Assim, por exemplo, quando se pretende efectuar uma rotação pura basta igualar as coordenadas do vector de translação a zero. Quando tal acontece a matriz de transformação fica assim com o seguinte aspecto:

$$Rot(\phi) = \begin{bmatrix} \cos(\phi) & -sen(\phi) & 0 \\ sen(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{EQUAÇÃO 16}$$

Aspecto diferente terá a matriz de transformação T, quando se pretende uma translação pura do referencial. Em tal situação a sub matriz de rotação ficará na forma de matriz identidade, comportando-se assim como elemento neutro na multiplicação de matrizes. Quando tal acontece a matriz T terá então o seguinte aspecto:

$$Trans(p_x, p_y) = \begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{EQUAÇÃO 17}$$

3.2.2. Matriz de transformação no espaço

Depois de explicado o processo de localização de um objecto no plano vai agora proceder-se á apresentação dos conceitos relativos à localização no espaço. No plano foi usada a matriz de transformação de dimensão três para traduzir todas as translações e rotações de um objecto. No espaço também se vai utilizar a mesma matriz de transformação, contudo neste caso esta terá dimensão quatro. Este incremento de uma linha e coluna na matriz de transformação acontece pois agora o sistema de dois eixos do plano deu lugar a um sistema de três eixos no espaço, sendo assim a matriz de transformação T passa a ser representada pela seguinte expressão:

$$T = \begin{bmatrix} a & b & c & p_x \\ d & e & f & p_y \\ g & h & i & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equação 18

Sub-matriz de Rotação

O princípio de funcionamento da matriz de transformação no espaço é o mesmo que no plano. Quando se deseja que ocorra uma translação pura, basta que a sub matriz de rotação assuma a forma da matriz identidade, tal como foi apresentado na Equação 17. Com um referencial tridimensional, a sub-matriz de rotação assumirá também a forma da matriz identidade, contudo terá dimensão três, assim a matriz T terá a seguinte forma:

$$T = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

EQUAÇÃO 19

No que toca á sub-matriz de rotação, esta possui três variantes elementares dependendo sobre que eixo se pretende a efectuar a rotação. Sabendo que no espaço o sistema de eixos é definido por x, y e z, e tal como já atrás foi referido, pode haver uma rotação sobre cada um destes eixos, tal como é apresentado na Figura 12. Para calcular a sub-matriz de rotação é possível recorrer de novo ao exemplo do elo que roda em torno do sistema de coordenadas de onde se obtinha o sistema de Equações 11. No exemplo dado anteriormente o elo roda no plano, contudo no espaço o mesmo elo irá rodar em torno do eixo zz. Nesse caso pode-se deduzir que a sub matriz de rotação tem a seguinte forma:

$$Rot(z,\theta) = \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

EQUAÇÃO 20

Nos casos em que a rotação ocorre sobre o eixo xx ou sobre o eixo yy tem-se respectivamente que:

$$Rot(x,\theta)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\theta) & -sen(\theta)\\ 0 & sen(\theta) & \cos(\theta)\end{bmatrix} \qquad \text{EQUAÇÃO 21}$$

$$Rot(y,\theta)=\begin{bmatrix}\cos(\theta) & 0 & sen(\theta)\\ 0 & 1 & 0\\ -sen(\theta) & 0 & \cos(\theta)\end{bmatrix} \qquad \text{EQUAÇÃO 22}$$

Tal como no plano, no espaço quando se pretende efectuar uma rotação pura colocam-se as coordenadas do vector de translação igual a zero, tal como na Equação 16, contudo agora a sub matriz de rotação pode assumir três formas, Equação 20, Equação 21 ou Equação 22 dependendo sobre que eixo é efectuada a rotação.

3.2.3. Interpretação da matriz de transformação

Depois de descrito todo o processo de obtenção de uma matriz de transformação falta agora descrever qual é o seu significado. É possível encontrar quatro significados para uma matriz de transformação.

1– Como a transformação de um sistema de coordenadas em outro;

2– Como a descrição da origem e orientação do novo sistema de eixos segundo o referencial anterior;

3- Como a descrição do movimento de um objecto de um ponto de origem, segundo um referencial, para outro ponto, segundo outro referencial;

4- Como o cálculo de uma nova posição segundo um referencial, em relação ao referencial inicial;

A matriz de transformação serve então para descrever a nova localização de um determinado objecto segundo um novo referencial. Assim, se se pretender descrever o movimento de um objecto ao longo de um determinado intervalo de tempo podem ser

usadas várias matrizes de transformações para o efeito. O produto de todas as matrizes de transformação será também uma matriz de transformação, esta descreverá a nova localização da origem do referencial inerente ao objecto segundo o referencial inicial. O produto de todas as matrizes de transformação conterá ainda a orientação do último sistema de eixos segundo o referencial inicial. É ainda importante referir que a ordem pela qual as sucessivas matrizes de transformação são multiplicadas é muito relevante, tendo essa ordem que estar bem definida para a correcta determinação das sucessivas posições do objecto em movimento.

Já atrás foi referido que uma matriz de transformação é constituída por quatro vectores, representados pelas quatro colunas da matriz. Os três primeiros vectores representam a orientação do novo sistema de eixos segundo o referencial anterior, em quanto que o último vector irá descrever a localização da origem do novo referencial segundo o referencial anterior.

3.3.Cinemática de um manipulador

A cinemática é a ciência que trata do movimento de um manipulador não levando em linha de conta as forças e as massas envolvidas. Este estudo apenas envolve a posição, velocidade, aceleração e suas derivadas, ficando a cargo da dinâmica o estudo das forças subsequentes ao movimento do robot. [9] A cinemática de um manipulador pode ser descrita usando a cinemática directa e/ou cinemática inversa. A cinemática directa é a abordagem que fornece a resposta á questão: “Onde e como está orientado o elemento terminal do robot, segundo um valor angular presente nas várias juntas do manipulador”. A questão que a cinemática inversa tenta resolver é: “Qual a posição angular de cada junta para que o elemento terminal possua uma determinada orientação e localização espacial?”.

Enquanto a cinemática directa e a cinemática inversa relacionam a posição angular das juntas com a localização espacial do elemento terminal, existe um terceiro tipo de cinemática que trata do estudo das velocidades e acelerações para um determinado movimento. Este terceiro tipo de cinemática é chamado de cinemática diferencial. [9]

Para solucionar os problemas da cinemática directa e da cinemática inversa basta conhecer as relações matemáticas existentes entre as posições de cada elo. Para tal ter-se-á que adoptar um sistema de coordenadas por elo e também utilizar alguns conceitos de álgebra linear.

3.3.1. Constituição mecânica de um manipulador

Um manipulador industrial, tal como qualquer máquina, é formada por um conjunto de peças que quando montadas de forma correcta fazem com que a máquina desempenhe correctamente a função para a qual foi projectada. Um manipulador industrial, tal como se vê na Figura 14, é genericamente constituído por elos ou *links*, que são as partes rígidas do braço robótico desempenhando um papel semelhante ao dos ossos humanos. São também constituídos por juntas ou *joints*, que são por norma motores eléctricos, sendo estas que determinam todo o movimento do robot, desempenhando um papel semelhante ao das articulações no corpo humano. Na Figura 14 também é possível ver que cada elo liga ao elo seguinte através de um motor eléctrico - as juntas. Assim, as juntas conectam os elos e permitem a realização de movimentos de um elo em relação ao elo anterior.

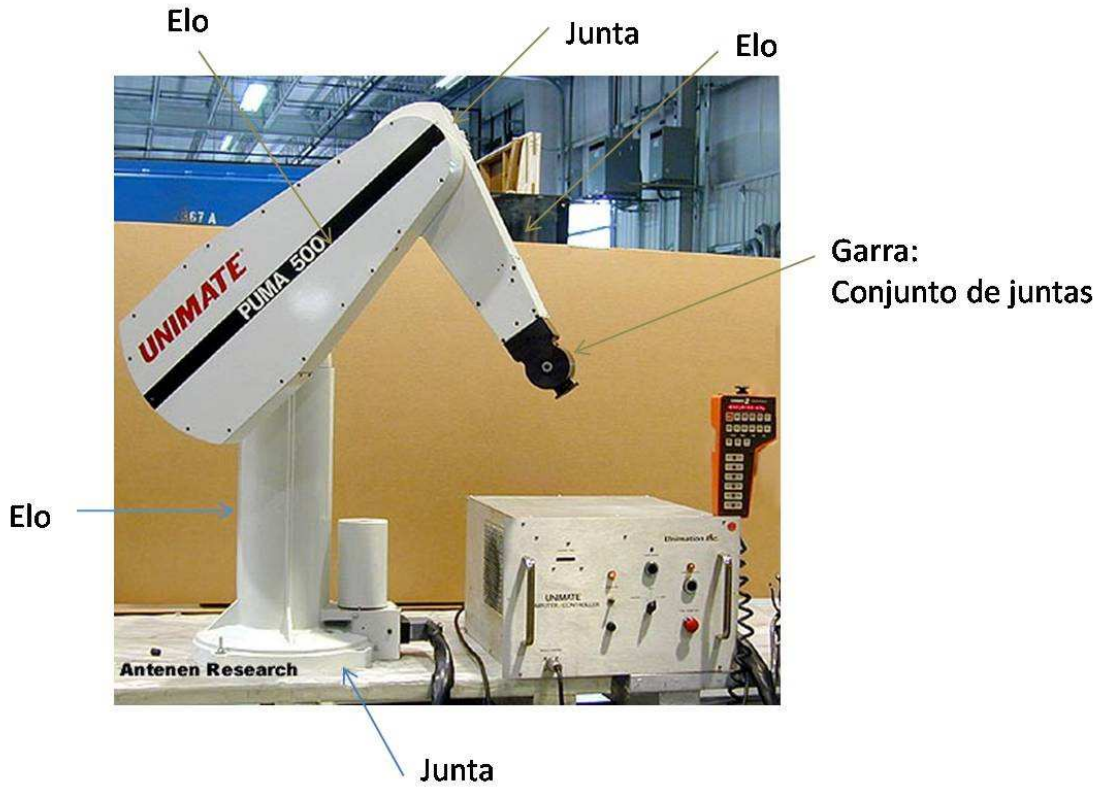

FIGURA 14 – EXEMPLO DE MANIPULADOR INDUSTRIAL

Tal como já foi dito, na Figura 14 existem vários elos e, como tal, existe a necessidade de esses mesmos elos serem numerados para que a diferenciação entre elos seja feita de forma simples e eficaz e para calcular a localização e orientação da garra do manipulador. Essa explicação será dada numa fase mais adiantada da dissertação. Os elos, ou *links*, de um manipulador são numerados por ordem crescente a partir da base do manipulador. Esta é designada como de elo 0, sendo o primeiro elo móvel designado de elo 1, os restantes elos são numerados de forma crescente. Na Figura 15 encontra-se a numeração de todos os elos do robot dado como exemplo na, Figura 14 no caso um modelo PUMA 560.

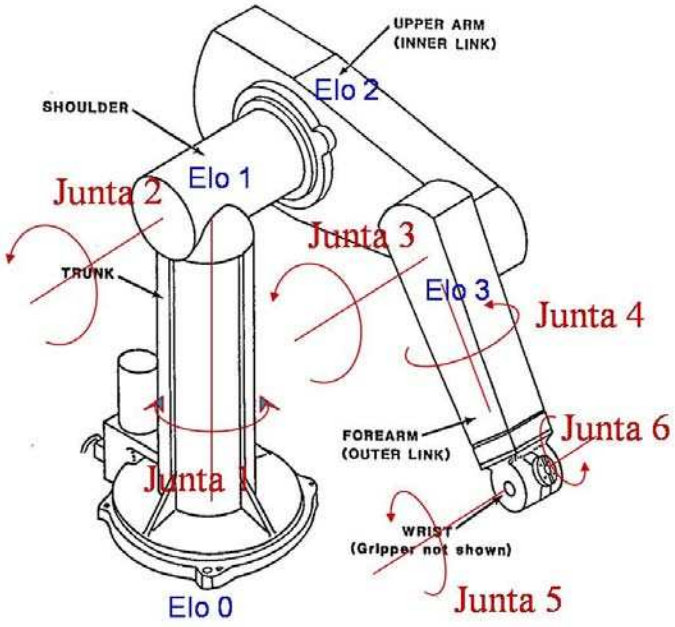

FIGURA 15 - NUMERAÇÃO DE ELOS E JUNTAS DO MANIPULADOR PUMA 560

Depois de apresentado o método de numeração de elos de um manipulador, vai agora ser apresentado o método de numeração de juntas, ou *joints*, de um robot, pois, tal como os elos, estas também têm que ser numeradas. As juntas são definidas por vectores no espaço 3D. Assim, a junta *i* é definida pelo vector no espaço sobre o qual o elo *i* aplica ou uma rotação ou uma translação em relação ao elo *i*-1. As juntas são numeradas a partir do elo 0 do robot, sendo a contagem das juntas iniciada no número 1. Tal como se pode ver na Figura 15, todas as juntas deste manipulador são rotacionais, logo cada uma delas representa um grau de liberdade do robot. Como este braço robótico tem um total de seis juntas, conclui-se então que este manipulador possui um total de seis graus de liberdade.

3.3.2. Parâmetros da cinemática

Tal como já foi referido, a cinemática de um manipulador trata da problemática de posicionamento do manipulador. Desta maneira, se se pretende localizar o elemento terminal esta mesma localização tem que ser referente a um qualquer referencial. Por norma, e ao longo de toda a dissertação, a localização do elemento terminal será sempre referente ao sistema de coordenadas presente na base do robot.

Para traduzir todas as translações e rotações que ocorrem ao longo do robot é necessário associar a cada elo um sistema de coordenadas. Desta maneira a relação geométrica entre elos pode ser traduzida através de uma matriz de transformação. O resultado do produto de todas as matrizes de transformação é uma matriz onde está representada a localização e a orientação do elemento terminal em relação ao referencial de origem.

No parágrafo anterior foi explicado o processo para determinar a posição e orientação do elemento terminal em relação ao referencial da base. Também foi referido que, para que tal fosse possível, era necessário atribuir um sistema de eixos a cada elo do manipulador. A atribuição dos vários sistemas de eixos não pode ser feita de forma arbitrária, sob pena de as matrizes de transformação traduzirem transformações geométricas erradas. Para atribuir correctamente um sistema de coordenadas a um elo é necessário levar em linha de conta dois factores: a própria geometria do elo; e os efeitos que este elo terá no elo seguinte do manipulador. Assim, para atribuir um referencial a um elo é necessário ter presente os conceitos de eixo da junta e parâmetros cinemáticos.

Eixo de uma junta é o eixo relacionando com a simetria do movimento inerente à própria junta, podendo este coincidir, ou não, com o eixo de um ou outro elo, podendo mesmo ser-lhe ortogonal. No exemplo que se tem vindo a tratar, o robot manipulador PUMA 560, tem apenas juntas do tipo rotacional com eixo de rotação perpendicular ao elo, sendo isso mesmo visível na Figura 15. Este eixo de junta faz parte do sistema de coordenadas associado a um elo, tendo-se convencionado que seria o eixo zz do mesmo. [3]

No que toca aos parâmetros cinemáticos, estes são quatro, estando divididos em dois grupos: os parâmetros dos elos e os parâmetros das juntas. Os parâmetros dos elos têm por missão definir a posição relativa e a orientação dos eixos da junta incidente no elo. Estes são, então:

Comprimento do elo (*link lenght*) – distância medida ao longo da normal comum entre os eixos das juntas *i-1* e *i*. Este parâmetro traduz o conceito de afastamento linear entre os eixos das juntas. Usualmente este parâmetro é definido por a_i (ver Figura 16);

Torção do elo (*link twist*) – ângulo de torção que é imposto pelo elo, sendo medido desde o eixo da junta anterior ao elo, *i-1*, até ao eixo da junta posterior ao elo, *i*. Usualmente é representado simbolicamente por α_i (ver Figura 16).

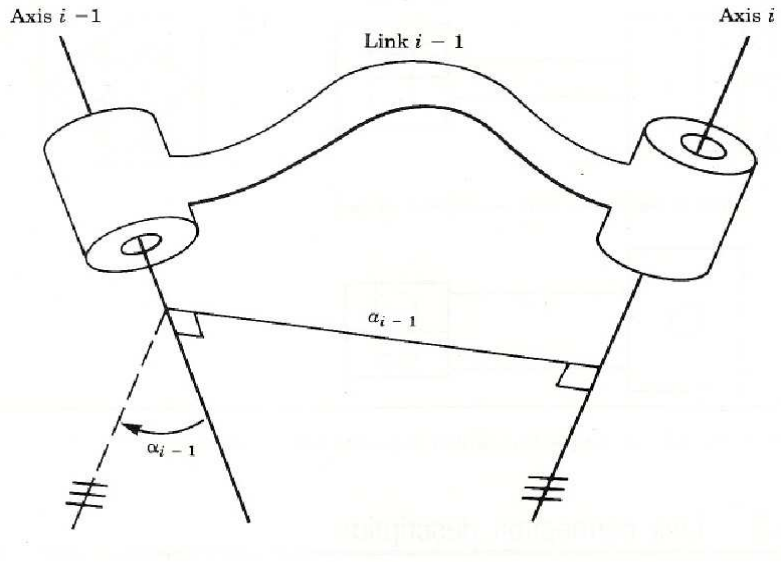

FIGURA 16 - PARÂMETROS DOS ELOS

Os restantes parâmetros cinemáticos, os parâmetros das juntas, são então:

Deslocamento de juntas (*offset*) – Distância entre elos medida ao longo do eixo da junta *i* entre as intercessões das perpendiculares com os eixos dos elos *i-1* e *i*. Sendo o eixo do elo o eixo colinear com a direcção deste. Usualmente este parâmetro é simbolicamente representado por d_i (ver Figura 17);

Ângulo de junta – ângulo definido entre o eixo de um elo *i*-1, e o eixo do elo seguinte, *i*. Por norma este parâmetro é simbolicamente representado por Θ_i (ver Figura 17).

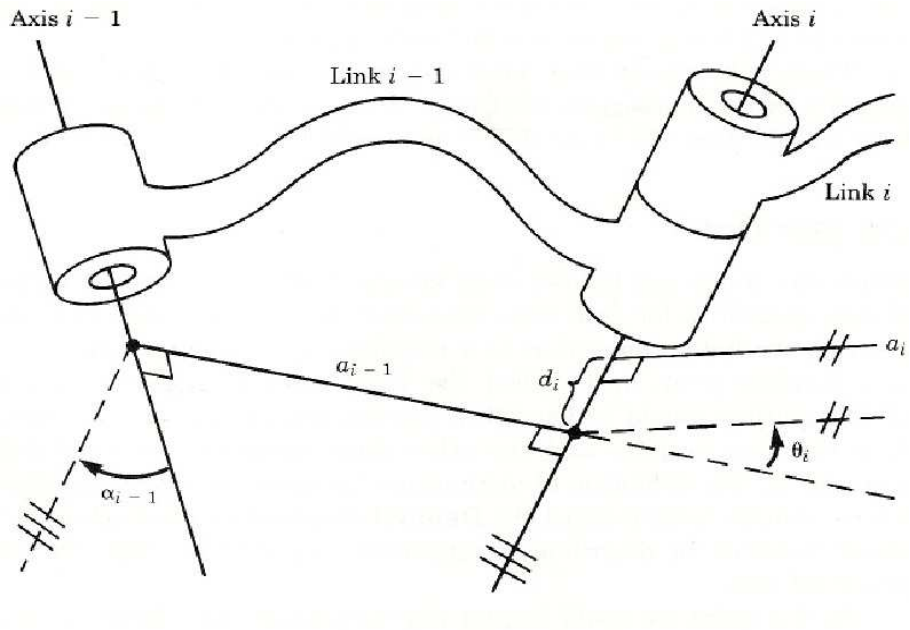

FIGURA 17 – PARÂMETROS DAS JUNTAS

Os quatro parâmetros cinemáticos anteriormente apresentados não são todos constantes ao longo do tempo. Se tal acontecesse as transformações geométricas seriam constantes, e consequentemente o manipulador não seria animado de qualquer movimento. No exemplo que tem vindo a ser apresentado, o manipulador PUMA 560, todas as juntas são do tipo rotacionais, assim todos os parâmetros cinemáticos de uma junta são constantes, excepto um, o ângulo de junta.

3.3.3. Transformação geométrica imposta por um elo

Depois de explicados os quatro parâmetros cinemáticos de cada elo de um manipulador, conclui-se que um elo *i* associado à sua junta *i*, realiza uma transformação geométrica. Essa transformação geométrica dá origem a um novo referencial *i*+1. Tal como já foi referido anteriormente, qualquer transformação geométrica pode ser decomposta em rotações e translações. Assim, a transformação geométrica pode ser decomposta nas seguintes operações elementares:

1 – Rotação em torno do eixo da junta *i*-1, num valor de Θ_i;

2 – Translação ao longo do eixo do elo (x_i), num valor do próprio comprimento do elo a_i;

3 – Translação ao longo do eixo da junta (Z_i), num valor do afastamento entre juntas d_i;

4 – Rotação do eixo da junta em torno do eixo do elo (x_i), num valor de α_i.

O produto das quatro transformações elementares dá origem a uma matriz com as mesmas dimensões, sendo essa matriz designada de A_i.

$$A_i = Rot(z,\theta)Trans(a_i,0,0)Trans(0,0,d_i)Rot(x,\alpha_i)$$

$$A_i = \begin{bmatrix} \cos(\theta_i) & -sen(\theta_i).\cos(\alpha_i) & sen(\theta_i).sen(\alpha_i) & l_i.\cos(\theta_i) \\ sen(\theta_i) & \cos(\theta_i).\cos(\alpha_i) & -\cos(\theta_i).sen(\alpha_i) & l_i.sen(\theta_i) \\ 0 & sen(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Os parâmetros cinemáticos apresentados e o método de obtenção da matriz de transformação, A_i, foram explicados à luz da convenção de Denavit – Hartenberg. [1] Existem, contudo, muitas outras convenções que dizem seguir a convenção de Denavit – Hartenberg, mas diferem em pequenos detalhes. Um exemplo é a convenção utilizada no livro "*Introduction to Robotics:* mechanics and control" [9], que sendo a de Denavit – Hartenberg difere na maneira de numeração dos sistemas de coordenadas e na forma de cálculo da matriz de transformação, A_i. Segundo este último autor a matriz de transformação é igual a:

$$A_i == \begin{bmatrix} \cos\theta_i & -sen\theta_i & 0 & a_{i-1} \\ sen\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -sen\alpha_{i-1} & -sen\alpha_{i-1}d_i \\ sen\theta_i sen\alpha_{i-1} & \cos\theta_i sen\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{EQUAÇÃO 23}$$

Durante esta dissertação serão usadas as duas convenções, sendo a primeira utilizada no caso do manipulador industrial existente nas instalações da empresa Selmatron, que o colocara à disposição deste trabalho, e a segunda foi aplicada na simulação efectuada ao manipulador industrial PUMA 760 (muito semelhante ao manipulador PUMA 560). A convenção usada pelo autor de "*Introduction to Robotics:* mechanics

and control" [9] foi usada no simulador uma vez que este foi construído tendo por base esta convenção. Em [9] o autor apresenta o cálculo da cinemática do manipulador PUMA 560 que é bastante idêntica á cinemática do manipulador PUMA 760. Desta maneira será usada a cinemática já calculada em [9] na aplicação construída. Mais informações sobre a cinemática deste manipulador pode ser obtida em [9].

3.3.4. Sistema de eixos ao longo da estrutura do manipulador

Tal como já foi referido, a cinemática de um manipulador trata a problemática de posicionamento do manipulador. Desta maneira, se se pretende localizar o elemento terminal, esta tem que ser referente a um qualquer referencial definido à partida. Assim, a posição e orientação do elemento terminal, durante toda a dissertação, será sempre calculado referente ao referencial presente na base do robot.

Para se saber a localização e orientação do elemento terminal é necessário obter a matriz de transformação que relaciona o referencial de origem e o referencial da garra do manipulador. Na grande maioria dos manipuladores industriais a base e a garra não se encontram ligados por um só elo, existindo quase sempre elos e consequentemente juntas intermédias. Desta maneira é necessário estabelecer sistemas de coordenadas referenciais intermédios. Consequentemente, em cada referencial intermédio será calculada uma matriz A_i que traduza as transformações geométricas ocorridas em cada elo. O produto de todas as matrizes A_i irá ser uma matriz de iguais dimensões à das que lhe deram origem, e irá traduzir a transformação geométrica ocorrida entre a base e a garra do manipulador. A matriz resultante é por norma designada de ${}^{R}T_{H}$, dependendo da literatura usada como referência. [3]

Até este ponto foi explicada a utilidade da atribuição de referenciais intermédios ao longo da estrutura do manipulador. Falta agora explicar a metodologia para a respectiva atribuição. Antes de se atribuir qualquer referencial é necessário numerar todos os elos e juntas presentes na estrutura do manipulador. Todo o procedimento de numeração de elos e juntas já foi previamente apresentado neste capítulo da

dissertação. Depois de todas as juntas e elos estarem numerados é necessário o seguinte procedimento para atribuição de sistemas de coordenadas:

Passo 1 – Estabelecer o sistema de coordenadas na base do manipulador. O sistema de coordenadas terá três eixos, x_0, y_0, z_0. O eixo z_0 terá que coincidir com o eixo da junta 1, podendo os restantes eixos, x_0, y_0, ser definidos da forma que se achar mais conveniente.

Passo 2 – Estabelecer todos os eixos z, de todos os referenciais de cada elo. Todos os eixos z, terão que ter a mesma direcção e mesmo sentido que todos os eixos de juntas do manipulador.

Passo 3 – Definir as origens de todos os referencias. Tal tarefa pode ser executada de duas maneiras diferentes: se o eixo z do sistema de coordenadas do elo anterior, z_{i-1}, se intercepta com o eixo z do sistema de coordenadas do elo pretendido, z_i, então esse local será a origem do referencial do elo; se o eixo z do elo actual e do anterior forem paralelos, então a origem do sistema de eixos do elo será a intercepção da normal comum a z_{i-1} e z_i, com o próprio eixo z_i. Para melhor se entender este conceito são apresentados alguns exemplos na Figura 18.

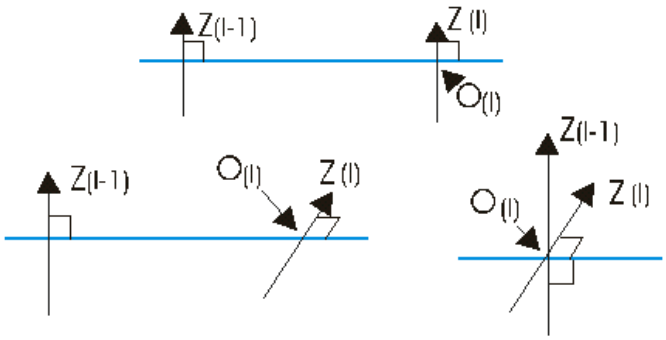

FIGURA 18 - EXEMPLOS PARA DEFINIÇÃO DA ORIGEM DE UM REFERENCIAL

Passo 4 - Definir o eixo x de todos os referenciais associados a cada elo. Tal pode ser feito de duas maneiras diferentes: caso o eixo z do sistema de coordenadas do elo anterior, z_{i-1}, se intercepta com o eixo z do sistema de coordenadas do elo pretendido, z_i, o eixo x_i do elo será o resultado do produto externo ($x_i = z_{i-1} \times z_i$); caso z_{i-1} e z_i sejam paralelos, então o eixo x_i será a normal comum a ambos os eixos. Para melhor entendimento deste conceito são apresentados alguns exemplos na Figura 19.

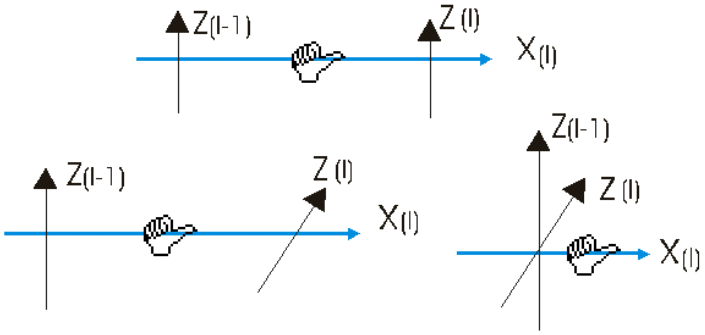

FIGURA 19 - EXEMPLOS PARA DEFINIÇÃO DO EIXO X DE UM REFERENCIAL

Passo 5 – Definir o y_i de cada sistema de coordenadas através do produto externo ($y_i=z_i \times x_i$). O mesmo produto externo poderá ser facilmente calculado se se recorrer à regra da mão direita, tal como ilustrado na Figura 20.

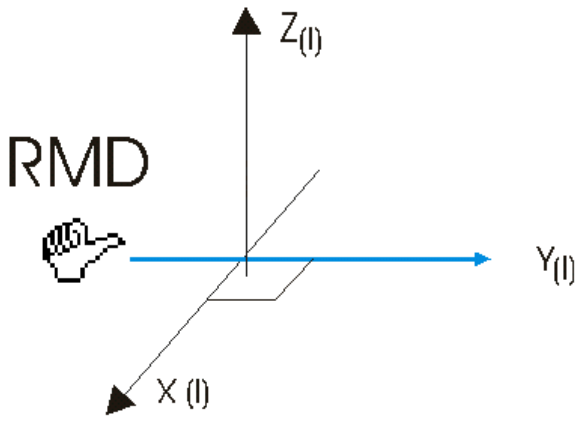

FIGURA 20 - EXEMPLO PARA DEFINIÇÃO DO EIXO Y DE UM REFERENCIAL PELA REGRA DA MÃO DIREITA

Passo 6 – Estabelecer o sistema de eixos do elemento terminal do manipulador. Neste referencial o seu eixo z, z_i, tem a mesma direcção e sentido que o eixo z, z_{i-1}, do referencial do elo anterior. No que toca ao eixo x, x_i, este é a normal entre z_i e z_{i-1}. Já o eixo y, y_i, é definido novamente através da regra da mão direita, pois é o resultado do produto externo ($y_i=z_i \times x_i$).

Depois de todos os referenciais estarem atribuídos falta apenas calcular os parâmetros da cinemática de cada elo para que se possa calcular a posição e orientação da garra do manipulador. Esse processo será exemplificado num ponto posterior desta dissertação.

3.3.5. Cinemática directa

Com o estudo da cinemática directa de um manipulador pretende-se obter uma resposta á seguinte questão: "Onde está o elemento terminal ou garra quando as suas juntam assumem um determinado valor?" Para se efectuar tal estudo é necessário definir dois espaços distintos: o espaço da juntas que simboliza o valor angular das várias juntas; e o espaço cartesiano que simboliza as coordenadas cartesianas do elemento terminal do manipulador. A cinemática directa é assim o estudo que permite obter o espaço cartesiano partindo do espaço das juntas.

Implementar a cinemática directa de um manipulador significa, portanto, determinar as relações que exprimem um ponto no espaço cartesiano. De forma genérica, o algoritmo que permite definir tais relações é o seguinte:

Passo 1 - Colocar o robot na posição zero, ou seja, colocar o robot na posição onde as várias variáveis das juntas assumam o valor de 0 - a referência absoluta;

Passo 2 - Atribuir a todos os elos do manipulador um referencial de acordo com o procedimento que foi anteriormente avançado;

Passo 3 - Verificar os quatro parâmetros da cinemática de cada elo do manipulador;

Passo 4 - Calcular as várias matrizes de transformação, A_i, para cada elo;

Passo 5 - Multiplicar as várias matrizes de transformação do manipulador de forma a obter a matriz de transformação que relacione a base com o elemento terminal ($^{R}T_{H}$);

Passo 6 - Através da matriz $^{R}T_{H}$ determinar a posição e orientação do elemento terminal.

No fim de aplicado o algoritmo da cinemática directa, a matriz $^{R}T_{H}$ irá exprimir a localização e orientação da garra do manipulador, contudo fica uma questão: sendo a matriz $^{R}T_{H}$ uma matriz quadrada, de dimensão quatro, como retiramos a informação da localização e orientação da garra? Em pontos anteriores desta dissertação já foi explicada a constituição de uma matriz de transformação, como é o caso da matriz $^{R}T_{H}$. Desta maneira, esta matriz é constituída por quatro vectores, $\vec{x}$, $\vec{y}$, $\vec{z}$ e $\vec{p}$. Tal como

já foi avançado anteriormente, os três primeiros vectores são referentes à orientação do elemento terminal, quanto que o último vector é referente à localização espacial, referente ao sistema de coordenadas da base do manipulador. Na Equação 24 está presente novamente a constituição simbólica de uma matriz de transformação.

$$ {}^{R}T_{N} = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$ **EQUAÇÃO 24**

No que toca ao vector de posição da garra, $\vec{p}$, este é constituído por três elementos p_x, p_y e p_z, sendo estes respectivamente as coordenadas segundo o eixo xx, yy e zz do sistema de coordenadas da base. Desta maneira o vector $\vec{p}$ pode ser descrito como o segmento de recta que une a origem do referencial do sistema de coordenadas da base com a origem do referencial da garra do manipulador. Para melhor ser entendido a Figura 21 apresenta uma ilustração que pretende exemplificar a localização dos vectores da matriz de transformação ${}^{R}T_{H}$.

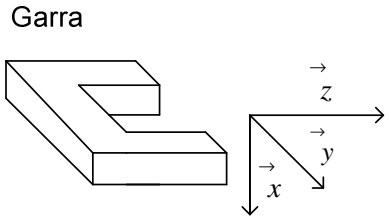

FIGURA 21 - ORIENTAÇÃO DO SISTEMA DE EIXOS DA GARRA DO MANIPULADOR

Os restantes três vectores da matriz de transformação $\vec{x}$, $\vec{y}$ e $\vec{z}$, são vectores que estão associados à orientação do elemento terminal do manipulador. O vector $\vec{x}$ é designado de vector de aproximação, tendo a direcção ortogonal ao vector $\vec{z}$. O vector $\vec{y}$ é designado de vector de escorregamento ou de deslizamento da mão, tendo a direcção do movimento de abertura ou fecho dos "dedos do manipulador". Finalmente, o vector $\vec{z}$ é designado por vector de ataque normal, tendo este a direcção normal aos "dedos" do manipulador.

3.3.5.1. Orientação da garra do manipulador

Tal como já foi referido atrás, na matriz de transformação, ${}^{R}T_{H}$, existem três vectores que pretendem traduzir a orientação espacial da garra do manipulador. Contudo existem mais formas para realizar essa missão. No segundo capítulo desta dissertação, foi referido que sendo a garra orientada segundo um referencial de três eixos, aos ângulos de rotação em torno de cada um dos eixos dava-se o nome de ângulos de Euler, sendo o seu significada alterado consoante a combinação de rotações usadas. Foi também referido que as combinações mais usadas no mundo industrial eram a *Roll – Pitch – Yaw* e também o denominado Tipo II dos ângulos de Euler. [3]

A combinação *Roll – Pitch – Yaw*, é uma combinação bastante imediata uma vez que se trata da rotação em relação a um referencial fixo segundo os eixos xx, yy e zz . Tal como já foi exposto anteriormente, a ordem pela qual são efectuadas várias rotações consecutivas é extremamente importante, não sendo este caso uma excepção. Desta maneira são efectuadas três rotações consecutivas: a primeira é efectuada sobre o eixo zz do referencial da Figura 22, sendo designada por *Roll*; a segunda é efectuada sobre o eixo yy do mesmo referencial, sendo designada de *Pitch*; a última rotação é efectuada sobre o eixo xx, do mesmo referencial que as anteriores rotações, sendo designada de *Yaw*.

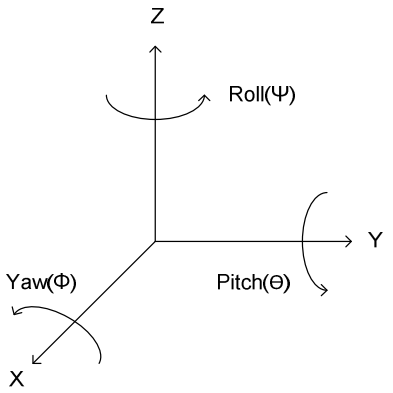

FIGURA 22 – ROTAÇÕES ROLL PITCH YAW NUMA GAARA DE UM MANIPULADOR

A matriz de transformação segundo este referencial é assim o produto de três matrizes rotacionais, sendo usualmente denominada de matriz RPY. Em baixo encontra-se a dedução e constituição da matriz RPY.

$$RPY(\phi,\theta,\varphi) = Roll(\varphi)Pitch(\theta)Yaw(\phi) = Rot(z,\varphi)Rot(y,\theta)Rot(x,\phi)$$

$$RPY(\phi,\theta,\varphi) = \begin{bmatrix} c_\phi s_\theta & -s_\phi c_\varphi + c_\phi s_\theta s_\varphi & s_\phi s_\varphi + c_\phi s_\theta c_\varphi & 0 \\ s_\phi c_\theta & c_\phi c_\varphi + s_\phi s_\theta s_\varphi & -c_\phi s_\varphi + s_\phi s_\theta c_\varphi & 0 \\ -s_\theta & c_\theta c_\varphi & c_\theta c_\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{EQUAÇÃO 25}$$

Depois de apresentada a combinação Roll-Pitch-Yaw vai agora ser descrita a outra variante para os ângulos de Euler - o Tipo II. É de ter em atenção que esta variação varia muito consoante o autor do livro ou documento. [3] Tal como na combinação Roll-Pitch-Yaw, também o Tipo II usa três rotações consecutivas. Contudo, neste tipo de variante existe inicialmente uma rotação em torno do eixo zz do referencial adoptado; a segunda rotação ocorre em torno do eixo yy, de novo referencial saído da rotação anterior; a terceira rotação ocorre novamente em torno do eixo zz, do novo referencial resultante da primeira rotação. Na Figura 23 encontram-se ilustradas as várias rotações que ocorrem segundo estas combinações de eixos.

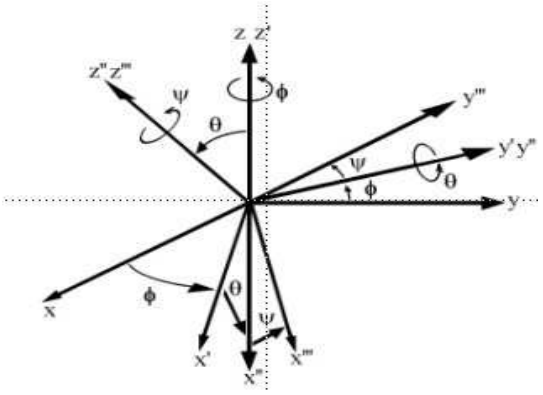

FIGURA 23 - VARIANTE DOS ÂNGULOS DE EULER TIPO II

A exemplo do que acontece com a combinação dos ângulos de Euler apresentada, também a matriz de rotação desta variante dos ângulos de Euler é decomposta em três rotações elementares, contudo neste caso as rotações ocorrem

segundo eixos diferentes. Desta maneira, a matriz de rotação segundo esta variante é igual a:

$$Euler_{II}(\varphi,\theta,\phi) = Rot(z,\varphi)Rot(y',\theta)Rot(z',\phi)$$

$$Euler_{II}(\varphi,\theta,\phi) = \begin{bmatrix} c_\varphi c_\theta c_\phi - s_\varphi s_\phi & -c_\varphi c_\theta s_\phi - s_\varphi c_\phi & c_\varphi s_\theta & 1 \\ s_\phi c_\theta c_\varphi + c_\varphi s_\phi & -s_\varphi c_\theta s_\phi + c_\varphi c_\phi & s_\varphi s_\theta & 1 \\ -s_\phi c_\theta & s_\theta s_\phi & c_\theta & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{EQUAÇÃO 26}$$

Depois de se ter exposto todos os conceitos teóricos, acerca das duas combinações dos ângulos de Euler, vai agora ser apresentada toda a ferramenta que permite determinar os ângulos de Euler a partir dos elementos da matriz de transformação do manipulador.

A matriz de transformação global, ${}^{R}T_{H}$, pretende traduzir a transformação geométrica ocorrida no elemento terminal em relação ao referencial da base do manipulador. Desta maneira esta pode ser decomposta em duas operações elementares, uma translação e uma rotação.

Tal como já foi atrás referido, a matriz de transformação global possui quatro vectores, tendo o vector $\vec{p}$ a responsabilidade de traduzir a localização da garra do manipulador. Pode-se então dizer que da base do manipulador até à garra do mesmo ocorre uma translação associada ao vector $\vec{p}$.

No que toca à orientação da garra do manipulador, esta não é mais que uma rotação do referencial da base do manipulador, podendo esta ser calculada através dos ângulos de Euler. Para tal basta escolher uma das doze combinações possíveis existentes. Em seguida irá ser demonstrado o cálculo dos três ângulos de Euler a partir da combinação Roll-Pitch-Yaw, contudo a metodologia necessária para o cálculo a partir do Tipo II seria exactamente a mesma.

Depois do exposto, pode-se então dizer que a matriz de transformação global é igual a:

$$
\begin{aligned}
{}^{R}T_{H} &= Trans(p_x, p_y, p_z)Roll(\varphi)Pitch(\theta)Yaw(\phi) \\
{}^{R}T_{H} &= Trans(p_x, p_y, p_z)Rot(z,\varphi)Rot(y,\theta)Rot(x,\phi) \\
{}^{R}T_{H} &= \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\varphi + c_\phi s_\theta s_\varphi & s_\phi s_\varphi + c_\phi s_\theta c_\varphi & p_x \\ s_\phi c_\theta & c_\phi c_\varphi + s_\phi s_\theta s_\varphi & -c_\phi s_\varphi + s_\phi s_\theta c_\varphi & p_y \\ -s_\theta & c_\theta c_\varphi & c_\theta c_\varphi & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

EQUAÇÃO 27

A matriz de transformação global que é obtida através do produto das sucessivas matrizes de transformação dos vários elos é igual a:

$$
{}^{R}T_{N} = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

EQUAÇÃO 28

As matrizes presentes na Equação 27 e na Equação 28 pretendem traduzir a mesma transformação, a da localização e orientação da garra do manipulador, contudo são calculadas de maneira diferente. Se ambas as equações são iguais então pode-se dizer que:

$$
\begin{cases} x_z = -sen(\theta) \\ y_z = \cos(\theta)sen(\varphi) \\ x_y = sen(\phi)\cos(\theta) \end{cases} \Rightarrow \begin{cases} \theta = arcsen(-x_z) \\ \varphi = arcsen(\dfrac{y_z}{\cos(\theta)}) \\ \phi = arcsen(\dfrac{x_y}{\cos(\theta)}) \end{cases}
$$

EQUAÇÃO 29

Desta maneira estariam determinados os três ângulos de Euler. Contudo, se se fizer uma análise mais detalhada observa-se que quando $\theta = 90^o$ ou nas suas imediações duas das expressões apresentadas anteriormente irão dar um resultado errado uma vez que a função arco-seno tem um domínio de -1 a 1. Outro problema é a

periodicidade da função co-seno, podendo este facto levar a perda de precisão dos resultados obtidos. A solução para estes problemas passa por usar mais equações extraídas da igualdade entre a Equação 27 e 28. Através destas a possibilidade de erro será minimizada.

Recorrendo novamente à igualdade entre matrizes tem-se então:

$$\begin{cases} x_x = \cos(\phi)\cos(\theta) \\ x_y = sen(\phi)\cos(\theta) \\ x_z = -sen(\theta) \\ y_z = \cos(\theta)sen(\varphi) \\ z_z = \cos(\theta)\cos(\varphi) \end{cases} \Rightarrow \begin{cases} \theta = arcsen(-x_z) \\ \varphi = \arctan(\frac{y_z}{z_z}) \\ \phi = \arctan(\frac{x_y}{x_x}) \end{cases}$$

EQUAÇÃO 30

As equações baseadas na função arco-tangente são bem mais robustas que as baseadas na função arco-seno, contudo esta impõe um novo problema: este tipo de funções não distingue os quadrantes em que estão os ângulos. Se por exemplo $y_Y=z_z=-1$, sendo o seu quociente igual a 1 e consequentemente $\Psi=45^0$. Como se vê, o resultado que se devia obter seria 135^0 e não o obtido. Este erro resulta do período da própria função tangente, sendo prática corrente o uso da função atan2(x,y) em detrimento da função arco-tangente. A função atan2 é na mesma a função inversa da tangente contudo faz a distinção de quadrantes.

3.3.6. Cinemática inversa

No ponto anterior foi apresentado o procedimento necessário para saber a localização e orientação do elemento terminal sabendo de antemão a posição angular das várias juntas do manipulador. Este procedimento é relativamente simples, contudo o processo inverso a este, ou seja, calcular a posição angular das várias juntas do manipulador para uma dada posição e orientação da garra do robot, é bastante mais complexo. [9] A cinemática inversa procura determinar o conjunto de valores das juntas que se adequam a uma dada posição e orientação da garra do manipulador.

Assim, a cinemática inversa pode ser vista como o conjunto de processos que permitem determinar as funções inversas das expressões da cinemática directa. [9]

O desafio imposto pela cinemática inversa é bastante mais complexo do que o imposto pela cinemática directa, uma vez que nem sempre existe uma solução analítica, ou por vezes não existe mesmo solução. Outra fonte potencial de problemas é o facto de poderem existir múltiplas soluções para uma dada configuração da garra. Este problema acontece devido à redundância existente na grande maioria dos manipuladores, ou seja, para a mesma configuração da garra do robot poderem existir mais que uma configuração das várias juntas do manipulador. Isso mesmo está exemplificado na Figura 24.

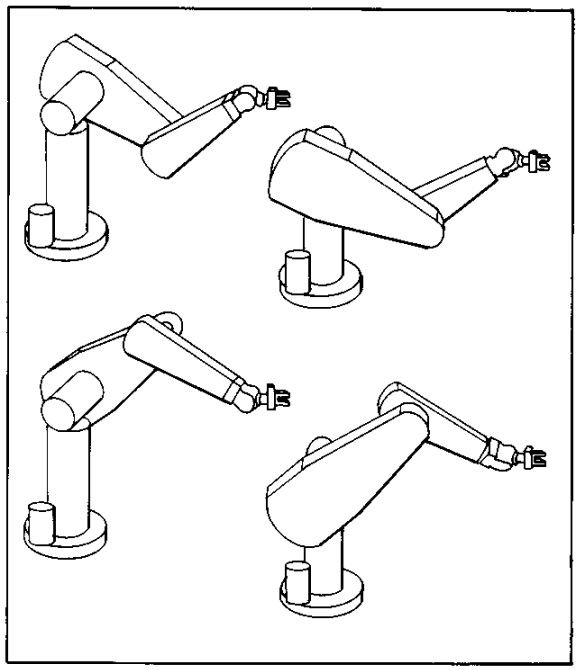

FIGURA 24 - REDUNCÂNDIA DO MANIPULADOR PUMA 560

Depois de apresentada a problemática da cinemática inversa e toda a complexidade inerente a este tipo de cinemática, falta agora apresentar os métodos que permitem obter o valor angular das várias juntas de um manipulador para uma determinada configuração do elemento terminal do mesmo.

Para se calcular a solução da cinemática inversa existem duas estratégias completamente distintas, a chamada forma fechada e a aproximação numérica. A aproximação numérica tem uma natureza iterativa e, como tal, é bastante mais

morosa que a forma fechada, sendo geralmente esta última a estratégia mais usada para o cálculo da cinemática inversa de manipuladores. [9] Desta maneira apenas irá ser apresentada esta estratégia de resolução nesta dissertação.

A estratégia denominada de forma fechada assenta em métodos baseados em expressões analíticas padrão, ou soluções de polinómios de quarto ou menor grau. Este último tipo de soluções parte do princípio de que não são usados quaisquer cálculos iterativos para calcular a solução do polinómio. Dentro da forma fechada existem dois métodos diferentes para o cálculo da cinemática inversa: o método algébrico e o método geométrico.

Quer o método algébrico, quer o método geométrico, são bastante similares, contudo diferem na abordagem feita ao manipulador. O método algébrico parte da matriz de transformação que relaciona o referencial da base com o referencial da garra do manipulador para, através de manipulação de algumas expressões, obter uma equação que permita calcular a posição angular de uma determinada junta, numa determinada configuração do manipulador. O método geométrico vê o manipulador como um só corpo e tenta decompor a geometria espacial do manipular em vários problemas de geometria planar. As soluções obtidas neste último método nem sempre são simples, sendo necessário efectuar simplificações. Se se observar uma solução obtida através deste método, constata-se que geralmente aparecem várias funções seno e co-seno. Nestes casos usam-se substituições polinomiais tais como:

$$u = \tan\frac{\theta}{2}$$
$$\cos\theta = \frac{1-u^2}{1+u^2}$$
$$sen\theta = \frac{2u}{1+u^2}$$

EQUAÇÃO 31

Para uma explicação mais detalhada das diferenças e da sequencia de operações que estes métodos aplicam é aconselhada a leitura do capítulo 4 do livro "Introduction to Robotics". [9]

Em qualquer dos métodos apresentados anteriormente uma solução só é considerada válida se obedecer a determinadas condições. [3] Essas condições são três e são as seguintes:

1 – A localização do elemento terminal, para o qual se pretende calcular a posição angular das juntas, tem que se estar inserido no espaço de trabalho do manipulador. Entende-se espaço de trabalho como sendo o conjunto de todos os pontos que o robot consegue alcançar;

2 – A posição angular de uma junta não deve exceder o seu limite físico;

3 – Para que exista uma solução analítica num manipulador de seis eixos é condição necessária e suficiente que três eixos de juntas sucessivas se intersectem ou sejam paralelos, desde que estes se interceptem num ponto no infinito. [9] Esta condição é vulgarmente conhecida por condição de Pieper.

Embora até este ponto da dissertação o método algébrico tenha sido descrito como sendo um método de cálculo da cinemática inversa, na verdade este não o é. É sim um conjunto de vários métodos que partilham o mesmo conceito. Na literatura existem vários métodos sendo os mais conhecidos: transformações inversas; matrizes duais; quaterniões duais. Para além destes métodos existem já determinadas soluções padrão para certos tipos de manipuladores, como por exemplo o braço manipulador antropomórfico de três graus de liberdade.

Capítulo 4

4. Estrutura do sistema

Neste capítulo descreve-se o hardware que fora inicialmente considerado como caso de estudo do trabalho aqui apresentado, sendo o elemento principal um robot manipulador de seis graus de liberdade. Embora não tenha sido possível testar este equipamento em funcionamento devido a não terem sido adquiridos, em tempo útil, os drivers dos motores das juntas do manipulador, todo o trabalho desenvolvido (testado em simuladores) foi talhado para este equipamento específico (com o objectivo de o poder vir a controlar) pelo que se entende fazer todo o sentido apresentar neste ponto a descrição do hardware considerado ao longo deste trabalho.

4.1.Constituição do manipulador da dissertação

Até este ponto da dissertação foram introduzidos alguns conceitos teóricos necessários para um entendimento total e efectivo do tema da dissertação. Seguidamente irá ser apresentado todo o hardware que constitui o manipulador robótico, assim como a interligação entre todos estes elementos.

O manipulador desta dissertação possui seis graus de liberdade e, consequentemente, seis juntas. Todas as juntas deste manipulador são do tipo rotacionais, sendo estas constituídas por motores eléctricos do tipo DC Brushless. Uma imagem do manipulador da dissertação pode ser vista na Figura 25. Os motores que inicialmente e actualmente, se pretendem usar nesta dissertação são os motores da série D31 da empresa *MOOG*, sendo que durante o decorrer da mesma equacionou-se o uso de motores família AM3000 da empresa Beckhoff, por motivos que serão

posteriormente apresentados. De seguida será feita uma pequena exposição sobre os motores de corrente contínua sem escovas.

FIGURA 25 - MANIPULADOR DA DISSERTAÇÃO

4.1.1. Motor DC *Brushless* de ímanes permanentes

Um motor eléctrico é um dispositivo capaz de converter energia eléctrica em energia mecânica. Assim existem dois grandes grupos de motores: os motores de corrente contínua (DC) e os motores de corrente alternada (AC). Dentro dos motores DC existem quatro configurações de motor: excitação em série, excitação derivada, excitação composta e ainda de ímanes permanente. Quanto aos motores AC existem três grandes grupos: os motores Síncronos, Assíncronos e Universais. [1]

Ao contrário dos motores de escovas, os motores sem escovas, ou *Brushless DC Motors*, ou ainda *BLDC Motors*, são motores eléctricos onde a operação de comutação,

que era efectuada por escovas no primeiro tipo de motor, é neste tipo de motores efectuada por um circuito eléctrico de controlo.

Os motores de corrente contínua sem escovas são essencialmente constituídos por duas partes distintas: o rotor e o estator. O rotor é constituído por ímanes permanentes, que podem estar colocados no interior ou no exterior do estator. [10] No que toca ao estator, este é geralmente constituído por material ferromagnético juntamente com os respectivos enrolamentos. Pode-se observar na Figura 26 um corte transversal de um motor deste tipo, onde é visível a constituição do estator e também todo o circuito electrónico que um motor deste tipo tem que ter obrigatoriamente acoplado.

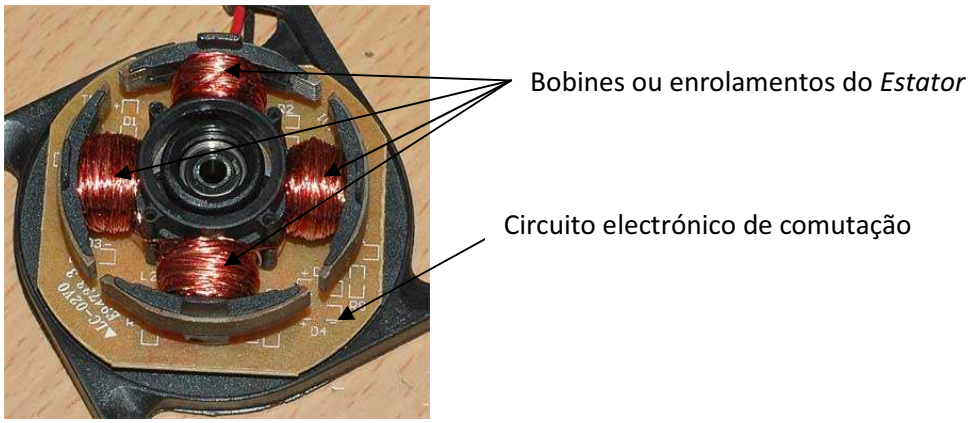

FIGURA 26 – CORTE TRANSVERSÃO DE MOTOR DC *BRUSHLESS*

A Figura 26 apresentada anteriormente é de um motor de uma pequena ventoinha de ventilação de um computador pessoal, contudo para aplicações industriais o estator é constituído por lâminas de chapas de aço empilhadas, com enrolamentos colocados nos entalhes e dispostos axialmente na periferia interna. [10] São também usados sensores de temperatura no interior dos motores para mediação da temperatura no interior do motor. Em função desta poder-se-á também medir a variação da grandeza eléctrica. [11] Os motores usados no manipulador estarão todos munidos com sensores termístores do tipo NTC.

O princípio de funcionamento deste tipo de motores é o seguinte: os enrolamentos do estator são alimentados com uma tensão DC, tendo esta alimentação que ter uma determinada sequência para que o motor rode no sentido e velocidade

pretendidos. Desta maneira são criados sequencialmente pares de pólos N-S no estator, atraindo assim os pólos S-N presentes no rotor, fazendo com que este entre em rotação. A tensão que surge aos terminais dos sensores de efeito de *Hall* é lida pelo circuito de comutação electrónico que determina, assim, a posição do rotor, fechando-se a malha de controlo do motor. Através desta malha de controlo o circuito electrónico é capaz de controlar três parâmetros do motor, são eles, a velocidade, o sentido de rotação e ainda o binário do motor. O controlo deste último parâmetro é feito através do fluxo magnético gerado pelos enrolamentos do *estator* consequentemente pela corrente que é fornecida aos mesmos enrolamentos. Assim quanto maior a corrente dos enrolamentos maior será o fluxo magnético produzido por estes e maior será o binário do motor.

Para se medir a posição angular do veio do motor usado nesta dissertação foi acoplado a este um sensor de posição. Esse sensor é um *resolver* que interage com o servo drive do seu respectivo motor. Embora este sensor só permita medir directamente a posição angular do veio do motor, pode-se calcular a velocidade do mesmo indirectamente, sendo esse trabalho feito num dispositivo externo ao motor, no caso o servo drive.

Este tipo de motores apresenta uma reduzida manutenção e longo tempo de vida útil, sendo também bastante silenciosos, com uma grande gama de velocidades e um elevado rendimento energético. Uma vez que os motores BLDC não usam escovas para fazer comutação, usando para tal um circuito eléctrico, têm um reduzido desgaste mecânico sendo responsáveis por uma produção muito baixa de interferências electromagnéticas ao contrário de outro tipo de motores. O uso deste circuito electrónico de comutação tem contudo o senão de tornar este tipo de motores mais dispendiosos relativamente aos motores que usam escovas.

O elevado número de vantagens que este tipo de motores têm, das quais se destacam a elevada precisão, reduzido desgaste mecânico e pequenas dimensões, fazem deles ideais para aplicações que exijam elevada fiabilidade. Motores BLDC são particularmente utilizados em áreas como a aviação, forças armadas, electrónica médica, equipamento informático e automação. Em jeito de conclusão falta apenas referir que este tipo de motores tem gradualmente vindo a conquistar cada vez mais

espaço de mercado, existindo variadíssimas aplicações onde se substituem motores DC com escovas por motores sem escovas.

Por fim falta apenas referir que as características dos motores utilizados nesta dissertação se encontram em anexo a este documento, mais propriamente no Anexo I.

4.1.2. Cinemática do manipulador da dissertação

Neste sub-capítulo da dissertação serão apresentadas todas as ferramentas desenvolvidas para o estudo da cinemática do manipulador. Foram programadas algumas aplicações em *Matlab* e *Labview* que permitem um melhor entendimento do funcionamento da cinemática do manipulador, quer da cinemática directa, quer da cinemática inversa.

Anteriormente já foi avançado em que consiste a cinemática directa de um manipulador, contudo não foi apresentado nenhum exemplo da sua aplicação. De seguida vai-se então proceder à aplicação de todos os conceitos da cinemática directa no manipulador da dissertação.

O manipulador a controlador possui seis juntas, consequentemente possui também seis sistemas de coordenadas a si associados. Na Figura 27 estão presentes todos os seis referenciais atribuídos ao manipulador, sendo que todos foram atribuídos consoante as regras previamente explicadas no capítulo anterior.

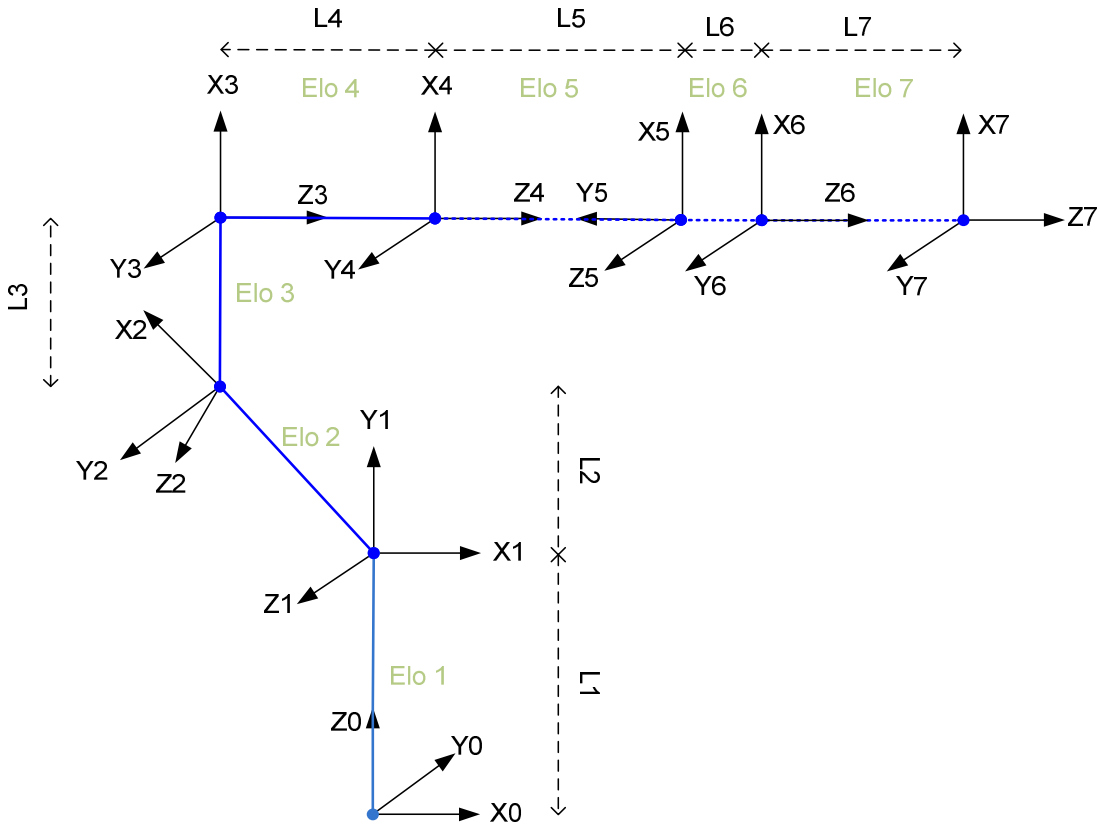

FIGURA 27 - SISTEMA DE EIXOS DO MANIPULADOR DA DISSERTAÇÃO

No parágrafo anterior foi dito que este manipulador seria definido por um total de seis referenciais existentes ao longo da sua arquitectura, contudo na Figura 27 podem-se observar mais do que esses seis referências. Isto acontece porque foi definido um referencial intermédio, o referencial com o terceiro índice. Isso mesmo também é traduzido pela tabela dos parâmetros da cinemática do manipulador, Tabela 2.

Depois de atribuídos os vários referências do manipulador falta agora apresentar os quatro parâmetros da cinemática referentes a cada referencial, na Figura 28 encontram-se os vários elos atribuídos ao manipulador. Conforme foi dito estes quatro parâmetros da cinemática fazem a tradução geométrica de todo o robot. Na Tabela 2 estão presentes os parâmetros da cinemática associados a cada referencial do manipulador.

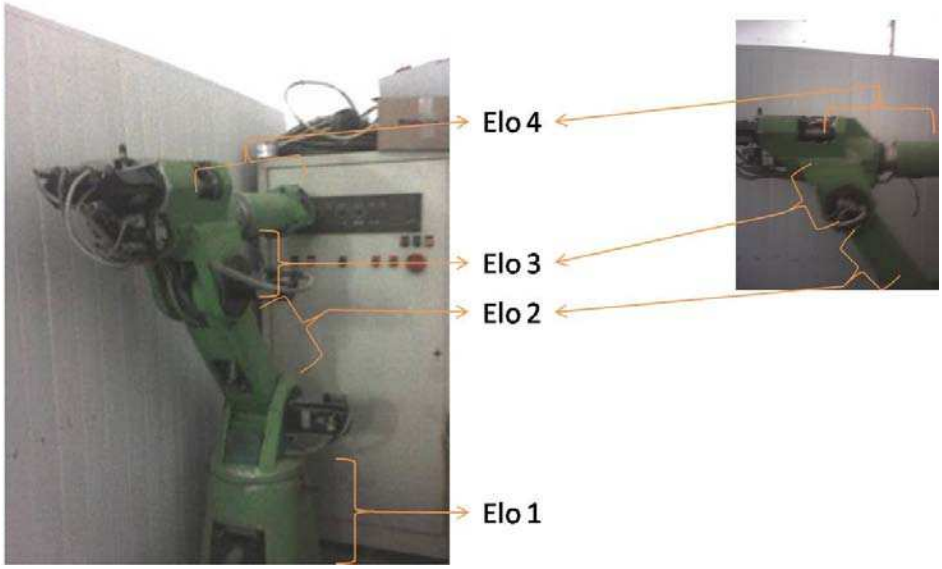

FIGURA 28 - ATRIBUIÇÃO DE ELOS AO MANIPULADOR DA DISSERTAÇÃO

TABELA 2- TABELA DE PARÂMETROS DA CINEMÁTICA DO MANIPULADOR

Parâmetros da Cinemática				
ELO	Angulo de Junta (Θ)	Deslocamento de Juntas (d)	Comprimento (a)	Torção do Elo (α)
1	Θ_1	L_1	0	0^0
2	Θ_2	0	L_2	0^0
3	Θ_3	0	L_3	90^0
4	0^0	L_4	0	0^0
5	Θ_4	L_5	0	-90^0
6	Θ_5	L_6	0	90^0
7	Θ_6	L_7	0	0^0

Na Tabela 2 anteriormente apresentada figuram os quatro parâmetros da cinemática associados a cada elo do manipulador da dissertação. Depois de determinados os parâmetros da cinemática serão agora apresentadas as matrizes de transformação associadas a cada elo.

$$A_1 = \begin{bmatrix} \cos(\theta_1) & 0 & sen(\theta_1) & 0 \\ sen(\theta_1) & 0 & \cos(\theta_i) & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

EQUAÇÃO 32

$$A_2 = \begin{bmatrix} \cos(\theta_2 + 270^0) & -sen(\theta_2 + 270^0) & 0 & L_3 * \cos(\theta_2 + 270^0) \\ sen(\theta_2 + 270^0) & \cos(\theta_2 + 270^0) & 0 & L_3 * sen(\theta_2 + 270^0) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ **EQUAÇÃO 33**

$$A_3 = \begin{bmatrix} \cos(\theta_3 - 45^0) & 0 & sen(\theta_3 - 45^0) & L_3 * \cos(\theta_2 - 45^0) \\ sen(\theta_3 - 45^0) & 0 & -\cos(\theta_2 - 45^0) & L_3 * sen(\theta_2 - 45^0) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ **EQUAÇÃO 34**

$$A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ **EQUAÇÃO 35**

$$A_5 = \begin{bmatrix} \cos(\theta_4) & 0 & sen(\theta_4) & 0 \\ sen(\theta_4) & 0 & -\cos(\theta_4) & 0 \\ 0 & 0 & -1 & L_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ **EQUAÇÃO 36**

$$A_6 = \begin{bmatrix} \cos(\theta_5) & 0 & sen(\theta_5) & 0 \\ sen(\theta_5) & 0 & -\cos(\theta_5) & 0 \\ 0 & 1 & 0 & L_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ **EQUAÇÃO 37**

$$A_7 = \begin{bmatrix} \cos(\theta_6) & -sen(\theta_6) & 0 & 0 \\ sen(\theta_6) & \cos(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & L_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ **EQUAÇÃO 38**

Devido ao facto de o produto de todas as matrizes originar uma matriz extremamente grande não será apresentada na dissertação. Contudo foi elaborada uma aplicação que permite visualizar esta mesma matriz. Para tal recorreu-se ao

programa *Matlab*, estando a referida aplicação contida no ficheiro 'Cinematica_Directa_Demonstracao.m'. Nesta aplicação são apresentados os elementos do produto de todas as matrizes. Os elementos da matriz são os apresentados na Equação 39.

$$ {}^{R}T_{N} = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$ **EQUAÇÃO 39**

Para testar se os parâmetros da cinemática da Tabela 2 estavam correctos, foi construído um simulador que permite analisar o movimento virtual do manipulador. O simulador foi construído em *Matlab* e é executado quando se corre o ficheiro 'simulacao.m'. Esse simulador permite também a visualização dos vários referenciais atribuídos aos sucessivos elos do manipulador. Quando o referido ficheiro é executado é pedido ao utilizador que introduza o número de iterações pretendidas para o movimento, isto é, se este pretender visualizar o movimento lentamente terá que introduzir um elevado número de iterações, se pelo contrário pretender um rápido movimento do modelo virtual do simulador terá que introduzir um baixo número de iterações. De seguida, o utilizador deve introduzir a posição angular, em graus, pretendida para as sucessivas juntas do manipulador.

Além do simulador apresentado anteriormente, foi também programada uma pequena aplicação em *Matlab* que permite calcular a localização da garra do manipulador, em função da posição angular das sucessivas juntas do manipulador. A referida aplicação está contida no ficheiro, 'Cinematica_Directa_Calculo.m', e permite obter valores numéricos das três coordenadas do vector de localização do elemento terminal do robot.

4.1.3. Cinemática do manipulador da dissertação

Depois de no ponto anterior se ter apresentado toda a cinemática directa do manipulador da dissertação, neste ponto irá ser apresentada toda a cinemática inversa

do manipulador a controlar. Para estes efectuar todos os cálculos necessários utilizaram-se algumas soluções padrão presentes em [3].

Calcular a cinemática inversa de um manipulador de forma analítica pode ser algo bastante complexo, podendo esta tarefa ser até impossível. Para que tal não aconteça, grande parte dos manipuladores utilizados nos dias de hoje, incluindo o da dissertação, usam o que se chama de punho esférico. O punho esférico faz com o eixo das três últimas juntas de um manipulador se interceptem num único ponto. Segundo a condição de *Piper*, apresentada no capítulo anterior, a cinemática inversa de um manipulador que apresente três juntas consecutivas, onde os eixos das respectivas juntas se interceptem num ponto, terá solução.

Manipuladores que possuam um punho esférico obedecem então à condição de Piper, e desta maneira as três últimas juntas do manipulador estarão associadas à orientação do punho, enquanto as três juntas iniciais estão associadas à posição do punho. Desta forma o problema da cinemática inversa pode ser dividido em dois: problema da posição, associado às três primeiras juntas; e o problema da orientação associado à orientação das três últimas juntas.

Depois de explicado o problema, irá ser explicado o procedimento para obtenção de todas as expressões matemáticas que permitem calcular a posição angular das várias juntas do manipulador em função da posição e orientação pretendida para a garra do manipulador. De seguida serão apresentados todos os cálculos teóricos efectuados para a obtenção das referidas equações.

Para o calcular as expressões das três primeiras juntas recorreu-se à matriz de transformação associada aos elos um, dois, três e quatro, sendo assim:

$$ {}^{0}T_{3} = A_{1} * A_{2} * A_{3} * A_{4} $$

$$ {}^{0}T_{4} = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Como apenas se pretende calcular as expressões que determinam as posições angulares em função da localização da garra, então a sub-matriz de orientação da matriz anterior não será considerada, sendo necessário e suficiente o vector de posição dessa mesma matriz. Dessa maneira o vector de posição é constituído por:

$$p_x = \cos(\theta_1).\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2}) + L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2}) - L_2.sen(\theta_2+\frac{\pi}{4}) \right]$$
$$p_y = sen(\theta_1).\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2}) + L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2}) - L_2.sen(\theta_2+\frac{\pi}{4}) \right]$$
$$p_z = -L_4.\cos(\theta_2+\theta_3+\frac{\pi}{2}) + L_3.sen(\theta_2+\theta_3+\frac{\pi}{2}) + L_2.\cos(\theta_2+\frac{\pi}{4}) + L_1 \quad \textbf{EQUAÇÃO 40}$$

No ficheiro *Matlab* 'Cinematica_Inversa_Demonstracao.m' encontra-se feito o cálculo da matriz anterior e do respectivo vector de posição. Os valores das três coordenadas do vector encontram-se já simplificados em relação aos devolvidos pelo *Matlab* no ficheiro anteriormente referido. Para tal foram utilizadas as fórmulas trigonométricas presentes no Anexo II da dissertação.

A posição angular do primeiro motor, θ_1 em função da localização da garra pode ser calculada da seguinte maneira:

$$\tan(\theta_1) = \frac{sen(\theta_1)}{\cos(\theta_1)}$$
$$\theta_1 = \arctan\left(\frac{sen(\theta_1)}{\cos(\theta_1)} \right)$$

Sabendo que:

$$\begin{cases} p_x = \cos(\theta_1).\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2}) + L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2}) - L_2.sen(\theta_2+\frac{\pi}{4}) \right] \\ p_y = sen(\theta_1).\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2}) + L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2}) - L_2.sen(\theta_2+\frac{\pi}{4}) \right] \end{cases}$$
$$\Rightarrow \frac{sen(\theta_1)}{\cos(\theta_1)} = \frac{p_y.\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2}) + L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2}) - L_2.sen(\theta_2+\frac{\pi}{4}) \right]}{p_x.\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2}) + L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2}) - L_2.sen(\theta_2+\frac{\pi}{4}) \right]}$$

$$\Rightarrow \theta_1 = \arctan\left(\frac{p_y}{p_x}\right)$$ **EQUAÇÃO 41**

A Equação 41 apresenta um problema ligado à determinação do quadrante do ângulo em causa. Por exemplo se, p_y=1 e p_x=1, o valor retornado pela Equação 41 será 45^0, contudo se p_y=-1 e p_x=-1, o valor retornado pela mesma expressão será na mesma 45^0, quando na realidade deveria ser -135^0. Este erro deve-se ao facto da função inversa da tangente ter uma periodicidade de 180^0. Para que tal não aconteça existe já em muitas linguagens de programação a função inversa da tangente que permite detectar o quadrante em que as coordenadas se encontram. Essa função tem o nome de atan2(), sendo apresentada na Equação 42.

$$\theta_1 = a\tan 2(p_y, p_x)$$ **EQUAÇÃO 42**

A posição angular do terceiro motor, θ_3 em função da localização da garra pode então ser calculada da seguinte maneira:

Da Equação 40 obtém-se p_x, sendo p_x^2 igual a:

$$p_x^2 = \left(\cos(\theta_1).\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2})+L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2})-L_2.sen(\theta_2+\frac{\pi}{4})\right]\right)^2$$

$$p_x^2 = \cos^2(\theta_1).\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2})+L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2})-L_2.sen(\theta_2+\frac{\pi}{4})\right]^2$$ **EQUAÇÃO 43**

Da Equação 40 obtém-se p_y, sendo p_y^2 igual a:

$$p_y^2 = \left(sen(\theta_1).\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2})+L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2})-L_2.sen(\theta_2+\frac{\pi}{4})\right]\right)^2$$

$$p_y^2 = sen^2(\theta_1).\left[L_4.sen(\theta_2+\theta_3+\frac{\pi}{2})+L_3.\cos(\theta_2+\theta_3+\frac{\pi}{2})-L_2.sen(\theta_2+\frac{\pi}{4})\right]^2$$ **EQUAÇÃO 44**

Da Equação 40 obtém-se p_z, sendo $(p_z - L_1)^2$ igual a:

$$(p_z - L_1)^2 = \left(-L_4.\cos(\theta_2 + \theta_3 + \frac{\pi}{2}) + L_3.sen(\theta_2 + \theta_3 + \frac{\pi}{2}) + L_2.\cos(\theta_2 + \frac{\pi}{4}) + L_1 - L_1 \right)^2$$

$$(p_z - L_1)^2 = \left(-L_4.\cos(\theta_2 + \theta_3 + \frac{\pi}{2}) + L_3.sen(\theta_2 + \theta_3 + \frac{\pi}{2}) + L_2.\cos(\theta_2 + \frac{\pi}{4}) \right)^2 \quad \textbf{EQUAÇÃO 45}$$

Através das Equações 43, 44 e 45, e recorrendo a algumas simplificações trigonométricas determinou-se a seguinte relação:

$$p_x^2 + p_y^2 + (p_z - L_1)^2 = L_3^2 + L_4^2 + L_2^2 - 2L_4L_2.\cos(\theta_3 + \frac{\pi}{4}) + 2L_3L_2.sen(\theta_3 + \frac{\pi}{4})$$

Recorrendo à expressão em anexo pode-se então calcular a expressão de θ_3 directamente se se considerar que:

$$k_1 = -2L_4L_2$$
$$k_2 = 2L_2L_3$$
$$k_3 = p_x^2 + p_y^2 + (p_z - d_1)^2 - L_3^2 - L_4^2 - L_2^2$$
$$\Rightarrow \theta_3 = 2.a\tan(k_2 \pm \frac{\sqrt{k_1^2 + k_2^2 - k_3^2}}{k_1 + k_3}) - \frac{\pi}{4} \quad \textbf{EQUAÇÃO 46}$$

Como se pode ver, a expressão que permite calcular θ_3 permite obter duas soluções possíveis pelo que esse facto introduz uma redundância na solução final.

Depois de se ter explicado o método de obtenção da posição angular do primeiro e terceiro motor será agira explicado o processo para obtenção da posição angular do segundo motor, θ_2 em função da localização da garra.

Como $r = \sqrt{p_x^2 - p_y^2}$ chega-se à seguinte igualdade:

$$r = \sqrt{\left[L_4.sen(\theta_2 + \theta_3 + \frac{\pi}{2}) + L_3.\cos(\theta_2 + \theta_3 + \frac{\pi}{2}) - L_2.sen(\theta_2 + \frac{\pi}{4}) \right]^2}$$

$$r - sen(\theta_2 + \frac{\pi}{4})\left[L_4\cos(\theta_3 + \frac{\pi}{4}) - L_3sen(\theta_3 + \frac{\pi}{4}) - L_2 \right] = \cos(\theta_2 + \frac{\pi}{4})\left[L_4sen(\theta_3 + \frac{\pi}{4}) + L_3\cos(\theta_3 + \frac{\pi}{4}) \right]$$

$$\cos(\theta_2+\frac{\pi}{4})=\frac{r-sen(\theta_2+\frac{\pi}{4})\left[L_4\cos(\theta_3+\frac{\pi}{4})-L_3 sen(\theta_3+\frac{\pi}{4})-L_2\right]}{L_4 sen(\theta_3+\frac{\pi}{4})+L_3\cos(\theta_3+\frac{\pi}{4})}$$ **EQUAÇÃO 47**

Se a p_z presente na Equação 40 se efectuar uma subtracção por L_1 obtém-se o seguinte:

$$p_z-L_1=-L_4.\cos(\theta_2+\theta_3+\frac{\pi}{2})+L_3.sen(\theta_2+\theta_3+\frac{\pi}{2})+L_2.\cos(\theta_2+\frac{\pi}{4})$$

$$p_z-L_1+\cos(\theta_2+\frac{\pi}{4})\left[L_4\cos(\theta_3+\frac{\pi}{4})-L_3 sen(\theta_3+\frac{\pi}{4})-L_2\right]=sen(\theta_2+\frac{\pi}{4})\left[L_4 sen(\theta_3+\frac{\pi}{4})+L_3\cos(\theta_3+\frac{\pi}{4})\right]$$

$$sen(\theta_2+\frac{\pi}{4})=\frac{p_z-L_1+\cos(\theta_2+\frac{\pi}{4})\left[L_4\cos(\theta_3+\frac{\pi}{4})-L_3 sen(\theta_3+\frac{\pi}{4})-L_2\right]}{L_4 sen(\theta_3+\frac{\pi}{4})+L_3\cos(\theta_3+\frac{\pi}{4})}$$ **EQUAÇÃO 48**

Se, agora se substituir a Equação 47 na Equação 48, e vise versa obtêm-se as seguintes igualdades:

$$A=L_4\cos(\theta_3+\frac{\pi}{4})-L_3 sen(\theta_3+\frac{\pi}{4})-L_2$$

$$B=L_4 sen(\theta_3+\frac{\pi}{4})+L_3\cos(\theta_3+\frac{\pi}{4})$$

$$\begin{cases} sen(\theta_2+\frac{\pi}{4})=\frac{[p_z-L_1]B+Ar}{A^2+B^2} \\ \cos(\theta_2+\frac{\pi}{4})=\frac{rB-[p_z-L_1]A}{A^2+B^2} \end{cases}$$

$$\Rightarrow \theta_2=a\tan 2(\frac{[p_z-L_1]B+Ar}{A^2+B^2},\frac{rB-[p_z-L_1]A}{A^2+B^2})-\frac{\pi}{4}$$ **EQUAÇÃO 49**

A Equação 49 pode ter quatro soluções possíveis mediante o valor atribuído a *r*. Se se somarem as quatro soluções possíveis para o cálculo de Θ_2, com as duas soluções possíveis para o cálculo de Θ_3, concluí-se que para um simples ponto existem seis soluções possíveis. Tal facto terá que ser levado em conta quando se fizer a programação da aplicação pretendida.

Foi feito um pequeno programa em *Matlab* que permite testar e confirmar que as expressões obtidas estavam correctas. No ficheiro 'Cinematica_Inversa_Calculo.m' encontra-se um pequeno programa que permite realizar tal tarefa. Inicialmente são introduzidos vários valores das posições angulares das várias juntas do manipulador. De seguida é calculada a localização da garra do manipulador mediante as várias posições angulares introduzidas. As coordenadas do elemento terminal são assim passadas para as funções de cálculo da posição angular das três juntas iniciais. Desta maneira é possível confirmar se os resultados das posições angulares obtidos são idênticos aos introduzidos.

Como última nota convém referir que apenas foi calculada a cinemática inversa para as três primeiras juntas uma vez que por especificação dos responsáveis da empresa, seria mais relevante efectuar apenas esta transformação.

4.2.Manipulador com sistema de comunicação *EtherCAT*

O manipulador da dissertação possuía aquando do início da dissertação motores, e servo drives dos respectivos motores, da marca *Moog*. A dissertação partia do princípio que os motores e servo drives estivessem operacionais, contudo quando se tentaram fazer alguns testes de funcionamento, estes revelaram estar inoperacionais (em todos os servo drives o led que indica falha por curto circuito ficou activo, sendo a solução indicada no manual a troca do mesmo). Devido a este percalço a empresa dona do material achou por bem encomendar motores e servo drives novos para o robot. Os novos motores, e respectivos servo drives, escolhidos são da empresa alemã Beckhoff. Os motores são da série *AM3000*, enquanto os servo drives são da série *AX5000*.

Os servo drives escolhidos possuem a particularidade de poderem comunicar com o seu controlador através do protocolo *EtherCAT*, sendo esta a principal razão que levou a optar-se por esta solução, desta maneira seria efectuado um controlo digital e usava-se um protocolo de comunicações o que leva a uma redução de custos de cablagem e uma maior integração a nível informático. A nova malha de controlo é

idêntica à já avançada no capítulo 2 desta dissertação, contudo a comunicação entre controlador e servo drive, e vise versa, é feita através do protocolo de comunicações *EtherCAT*. Para um melhor entendimento deste protocolo será de seguida feita uma breve descrição.

4.2.1. Protocolo de comunicações *EtherCAT*

O protocolo de comunicações *EtherCAT* (*Ethernet Control Automation Technology*) é um protocolo de comunicações industrial de alto desempenho e determinístico. Devido à sua característica determinista diz-se que é um protocolo de tempo real. O *EtherCAT* tem por base o protocolo *Ethernet*, e é normalmente utilizado em aplicações que exijam transmissão de dados e sincronização precisas. Este protocolo de alto desempenho é aberto, tendo sido publicado segundo a norma IEC 61158, e é largamente utilizado em projectos de máquinas e controlo de motores. (link sobre *EtherCAT*)

O protocolo *EtherCAT* obedece a uma arquitectura *Master*/*Slave*, sendo implementado sobre o padrão de cablagem Ethernet. Normalmente os *Masters* são controladores de tempo real ou autómatos, que recebem dos seus *Slaves* informações sobre o processo a controlar. Na Figura 29 estão representadas as três topologias suportadas por este protocolo, *Line*, *Tree* ou *Star*.

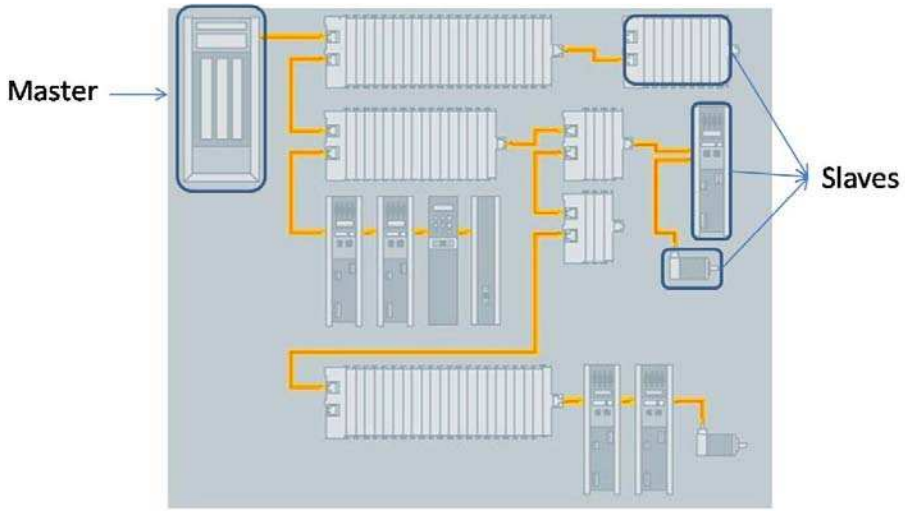

FIGURA 29 - TOPOLOGIAS DO PROTOCOLO *ETHERCAT*

Tal como já atrás foi referido, o protocolo *EtherCAT* tem por base um outro protocolo, o protocolo *Ethernet*, e como tal apenas usa a trama standard da norma IEEE 802.3. O transporte dos dados é feito directamente sob uma trama Ethernet, não sendo esta alterada em nada. *Master* e *Slave* comunicam entre si quando estão na mesma rede, podendo nessa situação pertencer ou não à mesma sub rede. Na primeira situação, a trama *EtherCAT* substitui directamente o protocolo *IP*, *Internet Protocol*, tal como está representado na parte superior da Figura 30. Na segunda situação, a trama *EtherCAT* virá encapsulada no protocolo UDP, *User Datagram Protocol*, tal como ilustra a parte inferior da Figura 30. [12]

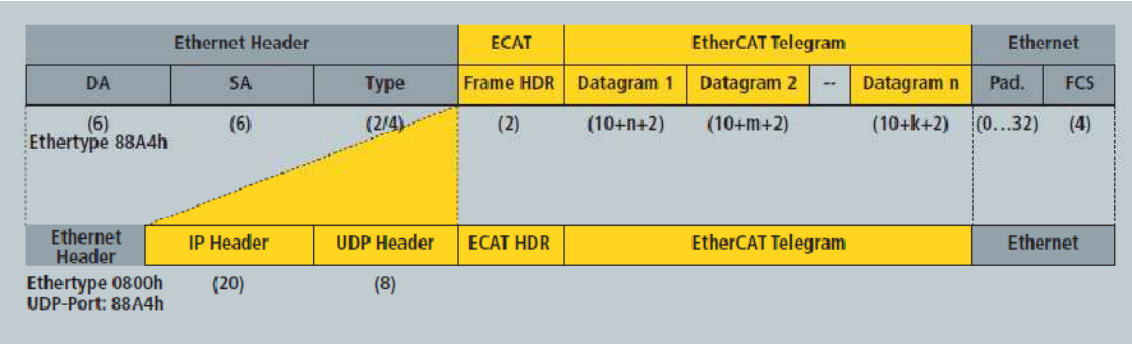

FIGURA 30 - TRAMA STANDARD DO PROTOCOLO *ETHERCAT*

No que toca à comunicação entre dispositivos, esta é feita sob a forma de objectos de processamento de dados, PDO, *Process Data Objects*. Assim cada *PDO* tem um determinado endereço, endereço esse que pode corresponder a um ou mais *Slaves*, dependo da informação que o *Master* pretende transmitir. A combinação de múltiplos *PDO* dá origem a um telegrama, sendo que uma trama *Ethernet* pode conter vários telegramas. Por vezes, em redes com muitos dispositivos podem ser necessárias várias tramas Ethernet para garantir que todos os telegramas requeridos são enviados aos seus destinatários.

O princípio de funcionamento deste protocolo pode facilmente ser entendido recorrendo à analogia de um comboio em movimento. O comboio é constituído por várias carruagens, tendo estas várias pessoas no seu interior. Assim, o comboio representa a trama *Ethernet* que está sempre em movimento, as várias carruagens representam os vários telegramas *EtherCAT*, e as pessoas os vários bits que constituem um *PDO*. Quando a trama *Ethernet* é enviada para a rede vai circular por todos os

Slaves sem parar. Quando uma trama passa por um Slave, este vê se existe algum *PDO* nela para si, se existir retira os dados que lhe são endereçados sem afectar o normal fluxo da trama pela rede. Quando a trama chega ao último *Slave* é remetida para o Master que inicialmente a enviou para a rede. No "caminho de regresso" a trama irá novamente passar por todos os *Slaves* da rede. Este princípio de funcionamento permite a troca de dados não só entre *Master* e *Slave*, mas também entre dois ou mais *Slaves* ou mesmo entre vários *Masters*. Todo o processo de transferência de dados do protocolo está ilustrado na Figura 31. [13]

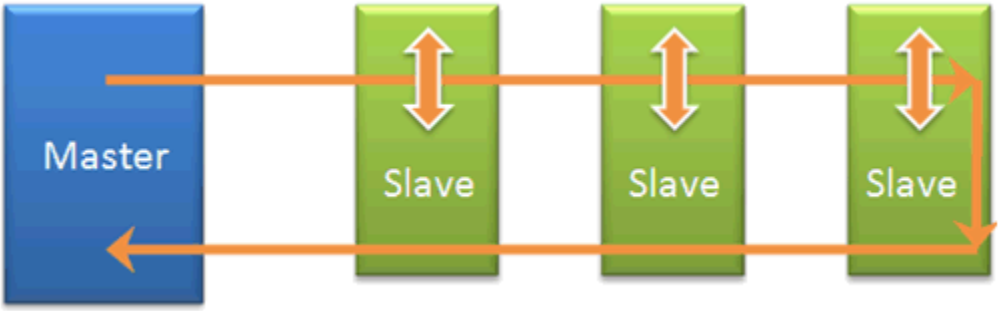

FIGURA 31 - ILUSTRAÇÃO DO PROCESSO DE TRANSFERÊNCIA DE DADOS EM *ETHERCAT*

Como já atrás foi referido, o protocolo *EtherCAT* é caracterizado por uma elevada performance. Para tal contribui em grande medida o facto de todo o processamento do protocolo ser feito ao nível do hardware, não estando refém do desempenho da CPU ou do tempo de execução da pilha do protocolo.

Outro importante factor de diferenciação do protocolo *EtherCAT* é o seu determinismo. Esta característica é especialmente importante para aplicações que exijam acções simultâneas ao longo de uma rede que pode ser bastante dispersa. Um exemplo da sua aplicação é a coordenação dos movimentos entre eixos. Para que a sincronização seja feita são utilizados etiquetas temporais (timestamps) que permite medir o tempo decorrido entre o envio e a recepção da mesma trama. Este método é também utilizado entre os vários nós da rede, permitindo assim uma sincronização precisa, com uma precisão de aproximadamente 1 micro segundo.

4.2.2. Sistema de testes com protocolo *EtherCAT*

Para se efectuarem testes, e para haver um estudo prévio de todo o material a adquirir, foi fornecido à Selmatron um kit de desenvolvimento. Esse kit encontra-se na Figura 32 e consiste num motor da série *AM3000*, um servo drive da série *AX5000*, cartas de comunicação de entradas e saídas, digitais e analógicas e ainda um controlador também ele *Beckhoff*, da série *CX*.

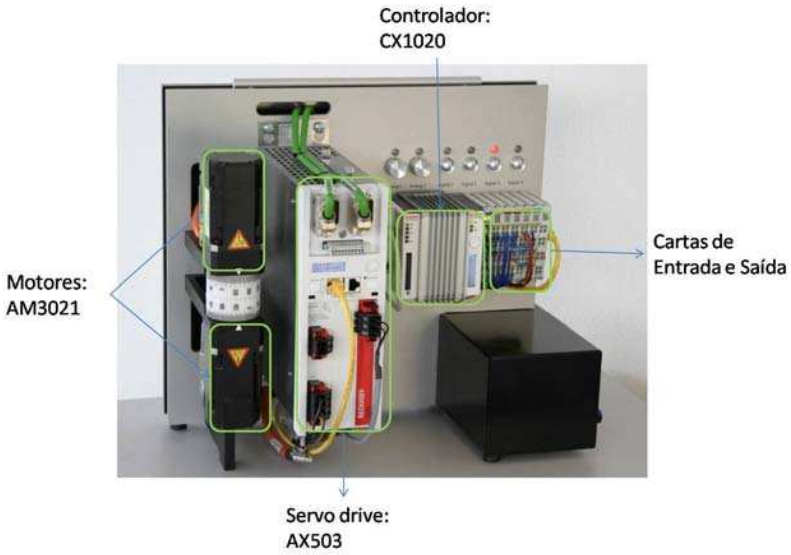

FIGURA 32 - *KIT* DE DESENVOLVIMENTO *BECKHOFF*

Uma vez que o controlador inicialmente previsto, *NI cRIO* 9002, não possui nenhuma forma de comunicar via protocolo *EtherCAT*, foi necessário escolher outro controlador que possui-se tal característica. A escolha recaiu sobre o controlador *NI PXI* 8176, uma vez que este possui também ele características de processamento de tempo real, uma arquitectura modular que permite a integração de diversos periféricos, entre eles uma placa de comunicações do protocolo *EtherCAT*. Na Figura 33 está presente uma imagem do controlador adoptado e da respectiva placa de comunicação com o protocolo *EtherCAT*.

FIGURA 33 - CONTROLADOR *NI PXI* 8176 E PLACA DE COMUNICAÇÕES *ETHECAT NI PXI* 8231

Sendo este controlador um verdadeiro computador, este possui características de tempo real, características essas que não seriam usadas se este usa-se um sistema operativo de base *Windows*. Desta maneira foi necessário instalar um sistema operativo de tempo real. O sistema operativo instalado foi a plataforma de tempo real disponibilizada pela *National Instruments*. Para se proceder a esta instalação foi necessário recorrer ao auxílio de um computador externo ligado em rede com o controlador.

Uma vez que o controlador *NI PXI* 8176 já existia nas instalações da empresa onde a dissertação decorreu, não foi necessário esperar que este estivesse disponível, o mesmo já não se passou relativamente à placa de comunicações por *EtherCAT*, *NI PXI* 8231. Uma vez que esta placa de comunicações era fundamental para toda a dissertação, foi necessário aguardar que esta viesse para se avançar com o software de toda aplicação. Durante o tempo de espera optou-se por se tentar elaborar um algoritmo que permitisse a comunicação com os drives através do controlador do kit de demonstração anteriormente mencionado. Na Figura 34 encontra-se o esquema de testes idealizado para fazer os testes mencionados.

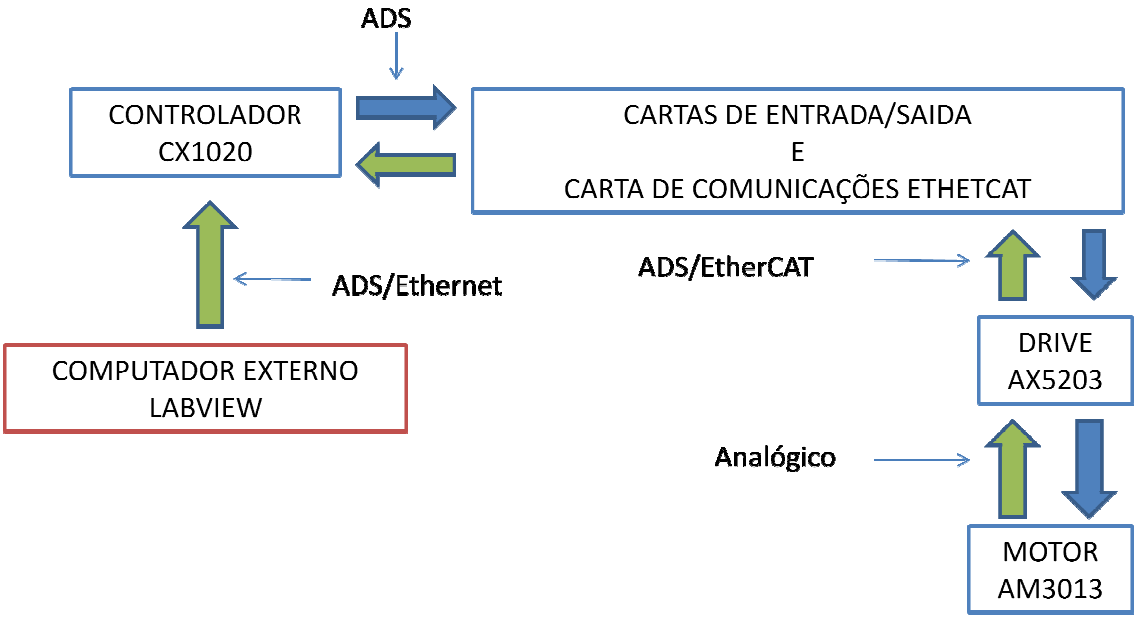

FIGURA 34 - SISTEMA DE TESTES DO PROTOCOLO *ETHERCAT*

Na figura anterior é visível que o acesso à rede *EtherCAT* é feito de forma indirecta através do controlador *CX*1020 utilizando para o efeito o protocolo de comunicações *ADS* e como protocolo de transporte o protocolo *Ethernet*. Uma vez que todo o material deste kit de desenvolvimento foi produzido pela empresa *Bekchoff*, foi necessário estudar e compreender o funcionamento da plataforma de configuração e programação desta empresa.

A plataforma de configuração e programação desta empresa tem o nome de *TwinCAT*, permitindo parametrizar todos os parâmetros configuráveis do material fornecido. Permite também a programação de aplicações que envolvam este tipo de material. Para realizar todas as configurações e transmissão/recepção de dados entre dispositivos é usado um protocolo proprietário da empresa, denominado de protocolo *ADS*. Para que a aprendizagem de todos estes conceitos fosse mais rápida e eficaz foi necessário frequentar uma acção de formação, na empresa Bresimar com um total de 36 horas, onde foi foram ministrados todos os conceitos já atrás mencionados.

No fim da referida acção de formação produziu-se uma pequena aplicação que permitia a actuação e comunicação com um autómato da série *BC* acoplado a cartas de entradas e saídas digitais, sendo esta aplicação totalmente programada em *Labview*. Desta maneira tornou-se possível a comunicação com dispositivos externos, através de

Labview, usando para tal o protocolo *ADS*. Na Figura 35 encontra-se uma imagem do painel frontal da aplicação e outra do diagrama de blocos produzido.

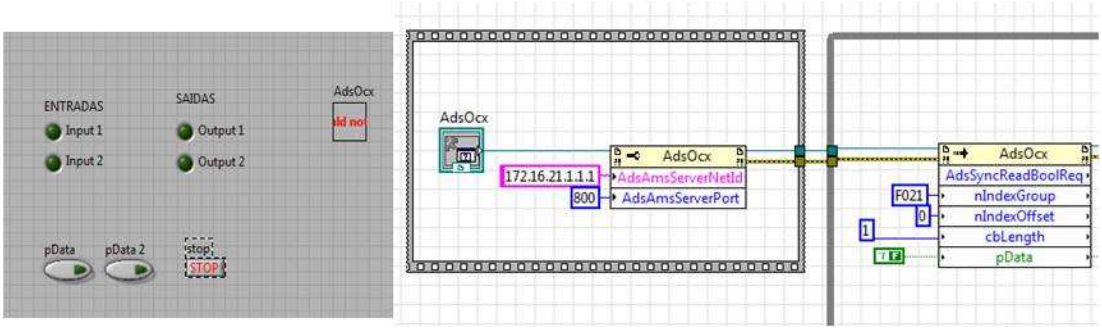

FIGURA 35 - PAINEL FRONTAL E DIAGRAMA DE BLOCOS DA APLICAÇÃO PARA COMUNICAÇÃO COM ÁUTOMATO BC 9050

Sem entrar em grande detalhe, a aplicação anterior utiliza um *OCX*, [14] que é uma ferramenta informática que permite a comunicação entre diferentes programas, para a ligação com o protocolo *ADS*. Além de se introduzir o endereço de rede *ADS*, em que o autómato se encontrava, foi também necessário introduzir o porto de comunicações a usar. Depois de explicada toda a parte de comunicações com o autómato, a parte de leitura e escrita das várias entradas e saída digitais das cartas de aquisição de sinal acopladas ao autómato, foi feita através de endereços de memória do autómato. Ou seja, para realizar uma simples leitura do estado de uma saída digital de uma determinada carta de aquisição de sinais, era lido o endereço de memória do autómato correspondente a essa saída digital. A escrita processava-se de forma semelhante, com a diferença de não se efectuar uma leitura, mas sim uma escrita no respectivo endereço de memória do autómato. O mapeamento de todos os endereços de memória foi feito previamente, utilizando para tal a plataforma *TwinCAT*.

Depois de estudada a forma de como o *Labview* pode comunicar através do protocolo *ADS*, passou-se aos testes propriamente ditos com o *kit* de desenvolvimento. Desta maneira pretendia-se colocar o motor a rodar através da plataforma *TwinCAT.* Tal como já foi avançado, o *kit* de desenvolvimento possuía um controlador da série *CX* que controlava todos os dispositivos. Desta maneira, utilizou-se um computador externo com a plataforma *TwinCAT* já previamente instalada, para se fazer a comunicação com o controlador, e consequentemente com todos os

dispositivos do *kit*. Essa comunicação foi possível através da ligação de um cabo *Ethernet*. Através da plataforma *TwinCAT* é possível controlar, configurar e programar todos os dispositivos da empresa *Beckhoff*. Desta forma utilizou-se esta plataforma para fazer rodar o motor acoplado ao servo drive. Através da plataforma *TwinCAT* foi possível observar a velocidade, aceleração e posição angular do veio do motor.

Depois de concluída a primeira tarefa, para colocar o motor do servo drive a rodar faltava agora elaborar uma aplicação em *Labview* que permitisse, numa fase inicial, a leitura de todos os dados enviados pelo servo drive, tal objectivo conforme se descreve a seguir foi atingido. Numa fase mais avançada do trabalho seria elaborada uma aplicação que permitisse não só a leitura dos dados, mas também o processamento e posterior envio de dados para o servo drive, este objectivo também ele foi atingido e descrito numa fase mais adiantada da dissertação. Tudo isto será feito numa fase inicial, utilizando para interface com a rede *EtherCAT*, o controlador *CX*, contudo posteriormente pretendesse elaborar uma aplicação capaz de comunicar directamente com o servo drive, utilizando para tal a placa de comunicações *NI PXI* 8231, neste último objectivo foram encontrados grandes problemas devido aos servo drives usado, de resto tais problemas serão descritos neste capítulo da dissertação.

Para a leitura de todos os dados enviados pelo servo drive para o controlador utilizou-se a mesma técnica que foi usada para a leitura e activação de das entradas e saídas digitais do autómato da *BC* 9050. Desta maneira, utilizou-se um *OCX* para que a comunicação através do protocolo *ADS* fosse possível. Posteriormente, recorreu-se ao *TwinCAT* saber quais os endereços de memória que haviam de ser lidos, e quais as variáveis aí presentes. Depois de se obter os endereços de memória foi elaborada uma aplicação em *Labview* para efectuar a leitura de todas as variáveis contidas nos endereços de memória recolhidos anteriormente. A aplicação tem o nome de 'Leitura_NC.vi' e permite a leitura da velocidade e posição do veio do motor acoplado ao servo drive. Todos os valores obtidos através deste método foram comparados com os que eram apresentados no *TwinCAT*, verificando-se serem iguais em ambas as aplicações. Na Figura 36 encontra-se uma imagem do diagrama de blocos (block diagram) da aplicação em questão.

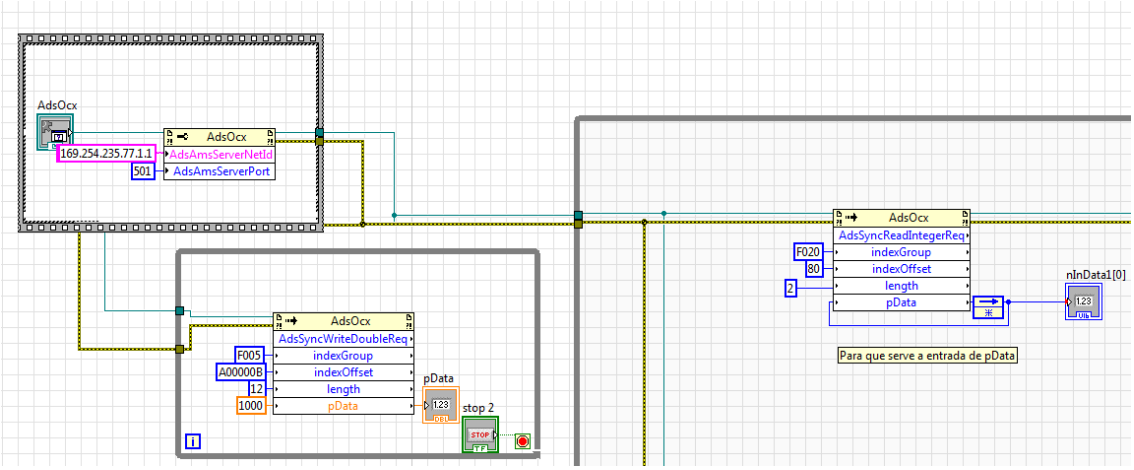

FIGURA 36 - DIAGRAMA DE BLOCOS DA APLICAÇÃO DE LEITURA DE ENDEREÇOS DE MEMÓRIA DO CONTROLADOR *CX* 1020

Através da aplicação anteriormente apresentada já se conseguia importar os valores de várias variáveis de toda a malha de controlo implementada pelo *TwinCAT*, faltava agora projectar um programa capaz de ler e escrever nos vários endereços de memória das variáveis de controlo do servo drive. A função utilizada para leitura de variáveis permite também a escrita, contudo esta revelou-se impossível devido a todo o kit ser controlado através do controlador *CX*. Desta maneira, quando se tentou colocar o motor em movimento, embora este tenha rodado ligeiramente, este não atingiu a posição pretendida. Para tal foi elaborado um programa extremamente simples de nome 'Escrita_AX500.vi'. A referida anomalia aconteceu porque, embora se fizesse a escrita no endereço de memória correcto, o controlador tinha uma tarefa interna (não controlável externamente) que voltava a escrever nesse endereço de memória, não deixando que o movimento fosse continuo. Este facto inviabilizou a utilização do kit de demonstração para se efectuar a actuação do, e comunicação com, o motor, pelo que se teve que aguardar pela chegada da carta de comunicações *EtherCAT, NI PXI* 8231.

Quando a carta de comunicações *EtherCAT* chegou procedeu-se à integração da mesma no sistema de controlo do manipulador. A Figura 37 ilustra a malha de controlo do manipulador pretenda.

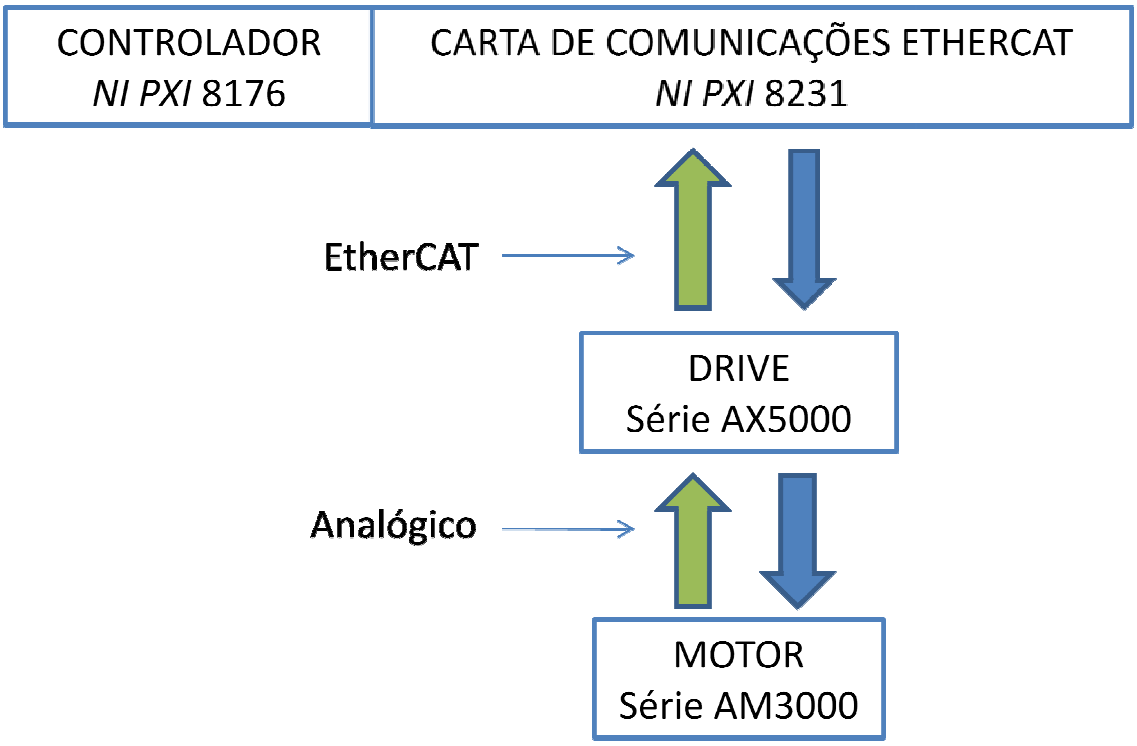

FIGURA 37 - MALHA DE CONTROLO IMPLEMENTADA EM *ETHERCAT*

Através da carta de comunicações *NI PXI* 8231 pretendeu-se que *Labview* comunicasse com o servo drive, e este posteriormente com o respectivo motor acoplado. Depois de se ter procedido à instalação da carta de comunicações, no sistema operativo de tempo real que corria no controlador de todo o sistema, tentou-se estabelecer comunicação com o servo drive. Para que a comunicação entre *Labview* e servo drive fosse possível, foi necessário importar um ficheiro XML que descrevesse todo o servo drive. Esse mesmo ficheiro é por norma fornecido pelo fabricante do dispositivo. Depois de se ter importado o ficheiro de descrição do servo drive tentou-se estabelecer uma comunicação efectiva, contudo tal não foi possível.

O servo drive escolhido, da série AX5000, utiliza como protocolo de transporte o protocolo de tempo real *EtherCAT*, tal como já foi avançado. O protocolo de transporte é apenas responsável pelo transporte de dados, não sendo da sua responsabilidade a compatibilização do tipo de dados a transmitir, ou seja, recorrendo a um exemplo simples, se duas pessoas estiverem a falar ao telefone, o protocolo de transporte só é responsável pelo transporte do diálogo entre participantes, ficando a cargo dos interlocutores a responsabilidade do estabelecimento de uma linguagem em

que ambos sejam capazes de comunicar. Desta maneira é usado um protocolo específico para que servo drive e controlador "falem a mesma língua", no caso do servo drive escolhido, série AX5000, é usado o protocolo *SERCOS*. Na Figura 38 está presente da forma como é encapsulado o protocolo *SERCOS* dentro protocolo *EtherCAT*. No ponto seguinte desta dissertação será feita uma breve apresentação do protocolo *SERCOS*.

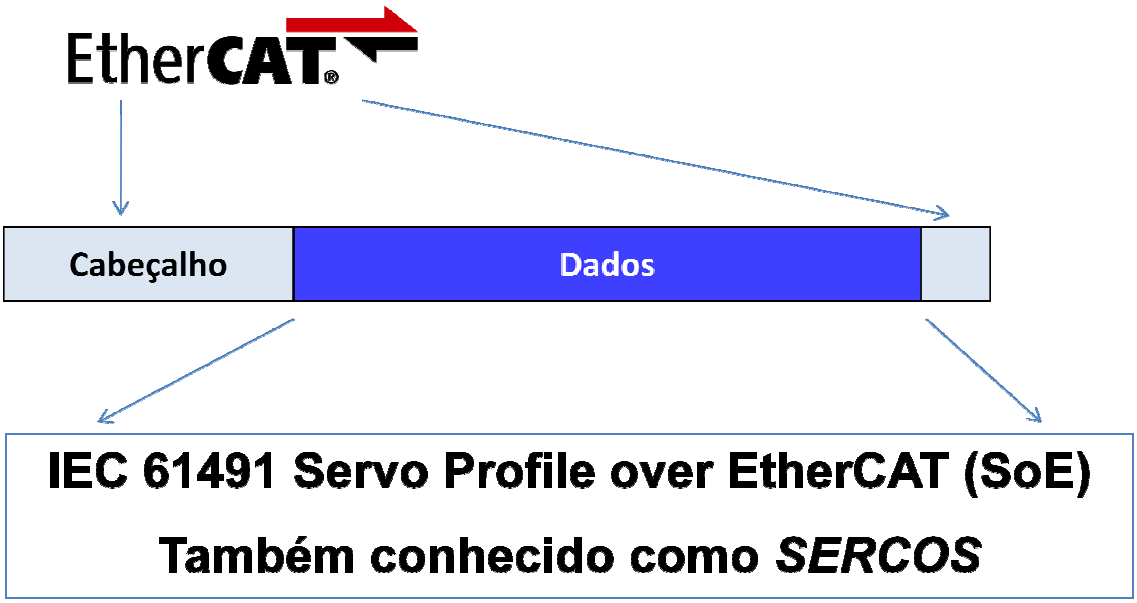

FIGURA 38 - ENCAPSULAMENTO DO PROTOCOLO *SERCOS* NO PROTOCOLO *ETHERCAT*

No parágrafo anterior foi mencionado que não se consegui estabelecer uma comunicação efectiva entre o servo drive e o controlador, contudo não foram apresentados os motivos de tal facto. Embora a comunicação *EtherCAT* fosse estabelecida, a comunicação via protocolo *SERCOS*, não era possível. Tal deveu-se ao facto de o *Labview* não possuir, no momento, as bibliotecas que permitam a utilização deste protocolo, e como este é fundamental para que controlador e servo drive falem a mesma "língua", não foi possível avançar mais com o desenvolvimento de uma aplicação onde se utilizassem servo drives que façam uso do protocolo *SERCOS*, nomeadamente o série AX5000. Perante este problema ainda foi equacionado o desenvolvimento em *Labview*, de uma biblioteca de funções que permitisse a integração deste protocolo, contudo depois de um estudo mais aprofundado, tal missão revelou-se impossível devido ao tempo necessário ser demasiado longo e devido a este protocolo ser de implementação prática bastante complexa.

4.2.3. Protocolo de comunicações *SERCOS*

O sistema de controlo de um manipulador, ou de qualquer dispositivo no mundo industrial, é constituído por vários sub-sistemas que necessitam de trocar informação entre eles. Para que a interligação dos diferentes componentes de um sistema de controlo seja feita de uma forma eficaz e controlada é necessária implementação de um protocolo que faça essa mesma gestão. Para fazer face a esta necessidade foi criado o protocolo *SERCOS*.

O protocolo *SERCOS* é especialmente indicado para equipamentos que possuam vários eixos de movimento e que exijam um elevado grau de precisão e coordenação de movimentos. Tipos de equipamentos que possuem estas necessidades são encontrados em máquinas de montagem, máquinas de embalagem, na robótica e em equipamentos de movimentação de materiais. O *SERCOS* (***SE****rial* ***R****eal-time* ***CO****mmunication* ***S****ystem*) é neste momento a interface globalmente padronizada para comunicação entre dispositivos de controlo de movimento, sendo classificado como a norma IEC 61491 e EN 61491. O *SERCOS* foi especialmente projectado para fornecer características de tempo real, alta performance de comunicação entre controlador e dispositivo de controlo do motor. [15]

Na década de oitenta a maioria dos sistemas de controlo usavam sinais em tensão analógicos para controlo de um motor. O controlador recebia do dispositivo de feedback um determinado sinal analógico que internamente era convertido para um determinado valor de posição ou velocidade. Após se processar o referido sinal analógico era calculada a saída a aplicar ao motor para que este atingisse uma determinada posição e/ou velocidade. A saída a aplicar era assim convertida num sinal analógico, de tensão, que era enviado para o servo drive do motor. Mediante a amplitude do sinal e a polaridade do mesmo, o servo drive actuava no motor fazendo com que atingisse a posição e/ou velocidade pretendida. O protocolo *SERCOS* veio, assim, assumir o papel desempenhado pelos sinais de tensão trocados entre os diversos componentes do sistema de controlo, fazendo com que toda a informação fosse transmitida digitalmente. Este protocolo veio fazer com que todos os componentes do sistema de controlo "falem todos a mesma linguagem". Na Figura 39

está presente uma pequena ilustração do sistema de controlo de uma máquina CNC implementado através do protocolo *SERCOS*.

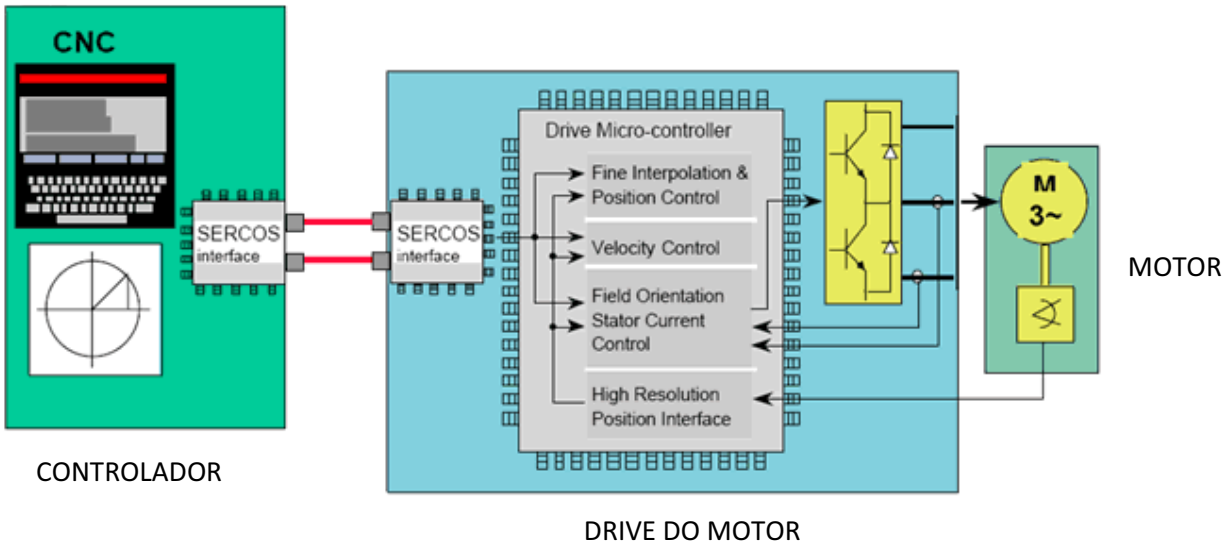

FIGURA 39 - EXEMPLO DE SISTEMA DE CONTROLO IMPLEMENTADO RECORRENDO AO PROTOCOLO *SERCOS*

Ao protocolo *SERCOS* são atribuídas cinco diferentes funções, sendo elas:

1 – Troca de dados entre controlador e drive do motor, mais especificamente, troca de comandos e valores actuais num intervalo de tempo reduzido;

2 – Garantir a sincronização de todos os componentes do sistema de controlo para que o equipamento execute movimentos precisos e coordenados nos vários eixos do mesmo;

3 – Capacidade de funcionar em três diferentes modos de controlo: controlo através do binário, velocidade, posição;

4 – Implementação de um canal de serviço não cíclico para transmissão de dados que não sejam imprescindíveis ao bom funcionamento do equipamento. Dados como parâmetros internos sinais de diagnóstico são transmitidos segundo este canal;

5 – Capacidade de integração no sistema de controlo de dispositivos de diferentes fabricantes, padronizando todos os parâmetros, formato de dados, comandos e dados de feedback do drive para o controlador.

Até ao momento existiram três gerações deste protocolo: o *SERCOS I*, *SERCOS II* e *SERCOS III*. As duas primeiras gerações usam para meio de propagação a fibra óptica. A única diferença existente entre estas duas gerações é a velocidade de transmissão de dados: enquanto que a primeira geração era capaz de transmitir dados a 2 ou 4 Mbits por segundo, a segunda geração era capaz atingir velocidades de transmissão de 2, 4, 8 ou 16 Mbits por segundo. Ambas as gerações usavam uma topologia do tipo anel e tinham uma arquitectura do tipo *master- slave*. Todo o sistema de controlo dos vários eixos do dispositivo formava um único anel, sendo o *master* o controlador da malha de controlo, e todos os outros drives são slaves do controlador, até um máximo de 254. Ao contrário da terceira geração deste protocolo, nas duas primeiras gerações só existia troca de informação entre *master* e *slave*, não existindo qualquer tipo de comunicação entre *slaves* da mesma rede. A terceira geração do protocolo *SERCOS* trocou de meio de comunicação - passou da fibra óptica para a rede Ethernet, passando desta maneira a usufruir de todas as vantagens desta.

Sem querer entrar em grande detalhe, neste protocolo todos os dados trocados entre dispositivos da mesma rede são trocados sob a forma de telegramas, existindo três tipos de telegramas: o MST, *Master Synchronization Telegram*, o AT, *Amplifier Telegram* e o MT, *Mater Telegram*. Inicialmente o master da rede envia um MST, para que todos os equipamentos fiquem sincronizados de forma a poderem calcular o intervalo de tempo em que devem proceder à aquisição de dados de *feedback*. Depois da sincronização feita, cada *slave* envia para o *master* toda a informação pré-estabelecida, sob a forma de um *Amplifier Telegram*. Depois de receber os *AT* de todos os *slaves* é enviado pelo controlador um telegrama sob a forma de *Master Telegram*, onde são transmitidas várias informações, como por exemplo comandos para um determinado *slave*. Cada *slave* sabe que posição do *MT* lhe pertence não lendo a que é destinada a outro *slave* da rede. [16]

As principais características deste protocolo são a sua elevada eficiência, características de tempo real, controlo e sincronização de vários eixos de movimento, e ser um standard comum a vários fabricantes de todo o mundo. Esta última

característica é especialmente importante uma vez que permite a integração de dispositivos de diferentes fabricantes no mesmo sistema de controlo.

4.3.Sistema de controlo usando servo drives da série MSD

Perante a impossibilidade de usar os servo drives da série AX5000, foi necessário escolher novos servo drives e novos motores. Desta vez optou-se por escolher equipamento que utilizasse como meio de comunicação entre controlador e servo drive diferentes níveis de tensão. Ou seja, optou-se pela utilização de dispositivo que usasse uma tensão com uma determinada amplitude e polaridade para transmitir toda a informação ao servo drive, sendo assim o sistema de comunicações baseado única e simplesmente em electrónica analógica.

Uma vez que o manipulador já possui desde o início motores da empresa *MOOG*, optou-se pela compra de servo drives da mesma empresa. Tal opção ficou a dever-se ao facto de a tecnologia dos motores ser praticamente idêntica deste há muitos anos para cá, e também por motivos económicos, pois desta maneira já não seria necessária a compra de novos motores. Os servo drives escolhidos foram da série MSD da já referida empresa. Além dos factores já apresentados, a escolha destes servo drives ficou também a dever-se ao facto de estes já virem preparados para que, no futuro, toda a parte de comunicações entre servo drive e controlador possa ser feita através de um qualquer protocolo, como por exemplo *Ethernet*, *EtherCAT* ou mesmo *CAN*.

Uma vez que se pretende que a comunicação entre controlador e servo drive seja feita através de electrónica analógica, foi necessário escolher um novo controlador. A escolha recaiu novamente sob o controlador *Compact Crio* da empresa *National Instruments*, devido a todas as características já apresentadas no primeiro ponto deste capítulo. A Figura 40 ilustra todo o sistema de controlo implementado com estes novos componentes.

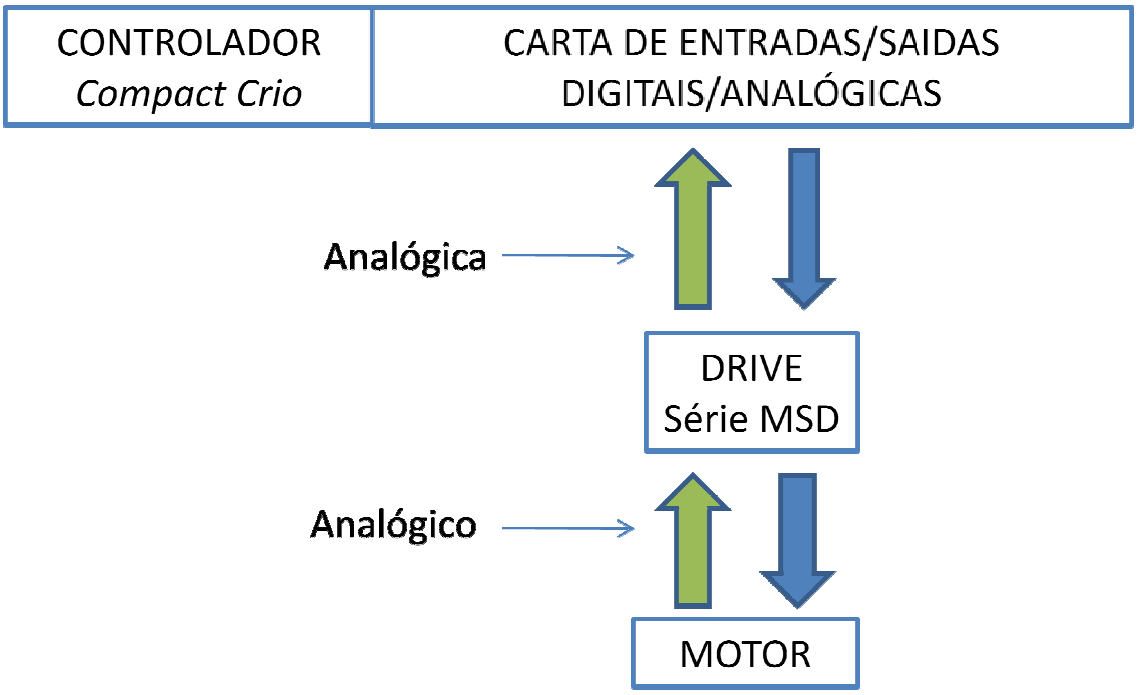

FIGURA 40 – SISTEMA DE CONTROLO COM CONTROLADOR COMPACT CRIO E SERVO DRIVE DA SÉRIE MSD

Tal sistema de controlo não foi ainda testado uma vez que os servo drives escolhidos não chegaram em tempo útil, tendo em conta a calendarização do trabalho considerado nesta dissertação. Durante todo o tempo de escolha e compra destes novos servo drives, por minha iniciativa e curiosidade pensei em desenvolver um simulador que fosse capaz de validar todos os cálculos já efectuados, e que me permitisse também aprofundar todos os conhecimentos necessário da linguagem de programação a utilizar na dissertação, *Labview*. De facto, tendo observado a morosidade inerente à aquisição de novos equipamentos de hardware, resolvi dar continuidade ao desenvolvimento da plataforma de interface e controlo dos motores do manipulador e, para testar este software, desenvolvi simuladores que substituíssem o hardware que, mais tarde, viria a ser implementado. Desta maneira elaborei várias aplicações que me permitiram ter um conhecimento mais assertivo e concreto de todas as problemáticas envolvidas na programação de uma aplicação, capaz de controlar este tipo de manipuladores. Tais aplicações serão apresentadas no próximo capítulo desta dissertação. Seguidamente será feita uma pequena apresentação de todo o material e software envolvido nesta dissertação.

4.4. Linguagem de programação *Labview*

O *Labview* (acrónimo para *Laboratory Virtual Instrument Engineering Workbench*) é uma linguagem de programação gráfica, habitualmente designada por linguagem G. Esta linguagem foi desenvolvida pela empresa *National Instruments*, tendo esta lançado a primeira versão em 1986 para computadores da empresa *Macintosh*. Devido à sua facilidade de programação, totalmente transparente a quem tenha os conhecimentos básicos de programação, e à sua elevada versatilidade, o *Labview* rapidamente se expandiu para outros sistemas operativos tais como o Windows, Linux e Solaris.

Actualmente o *Labview* é uma linguagem que está especialmente vocacionada para aplicações na área da automação industrial, instrumentação electrónica e controlo. O modelo de programação deste tipo de linguagem, fluxo de dados, confere-lhe vantagens em aplicações que possuam funções de aquisição de dados e posterior manipulação dos mesmos.

Os programas desenvolvidos em *Labview* são designados de VI's, Virtual Instruments, devendo-se este facto às elevadas semelhanças que os vários blocos constituintes de um VI apresentam com instrumentos, tais como osciloscópios ou multímetros. Uma aplicação desenvolvida em *Labview* pode ser constituída por apenas um VI ou por múltiplos VI's. Quando tal acontece é necessário que todos os VI's pertençam ao mesmo projecto sob pena de existirem erros durante a execução da aplicação. Tal como já foi dito anteriormente, todos os programas desenvolvidos têm o nome de VI, sendo estes constituídos por três elementos: *Front Pannel*; *Block Diagram*; e *Icon*.

O *Front Pannel* ou painel frontal representa toda a interface com o utilizador. É aqui que o utilizador interage com a aplicação sendo toda a parte de código propriamente dita ocultada do utilizador. Um exemplo de uma interface está visível na Figura 41.

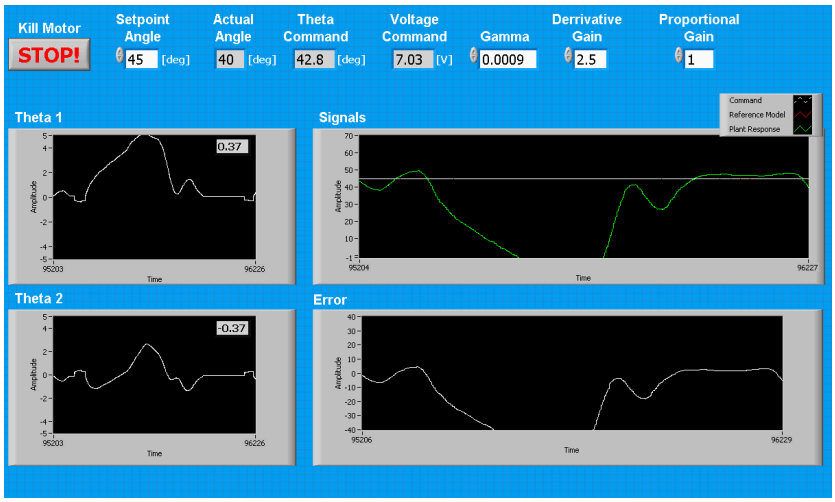

FIGURA 41 - EXEMPLO DE UM *FRONT PANNEL* OU PAINEL FRONTAL

O *Block Diagram* ou diagrama de blocos é o local onde o programador define toda a estrutura do algoritmo, usando para tal pequenos blocos de funções já existentes nas várias bibliotecas do *Labview*. Se estas não foram as mais adequadas para a aplicação pretendida o programador pode importar bibliotecas externas para o interior do *Labview*, podendo a partir desse momento usá-las livremente. Numa aplicação, o *Block Diagram* tem a aparência da Figura 42.

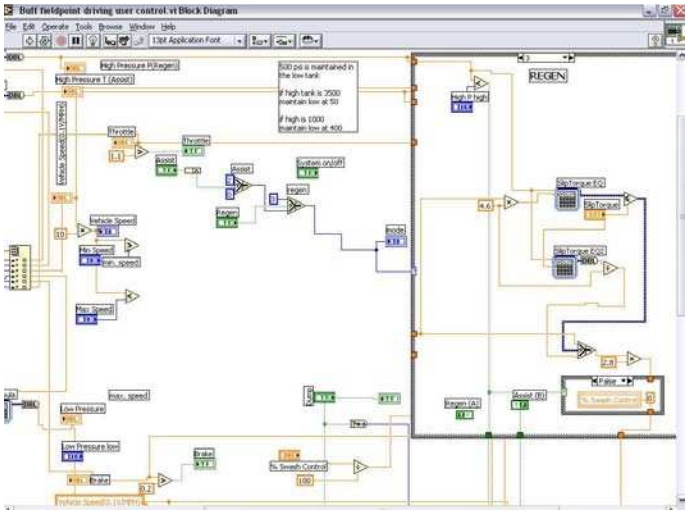

FIGURA 42 - APARÊNCIA DO BLOCK DIAGRAM OU DIAGRAMA DE BLOCOS

No que toca ao *Icon*, este identifica o VI quando este é usado como uma sub-rotina em outras aplicações. Um exemplo de um *icon* de um VI encontra-se representado na Figura 43.

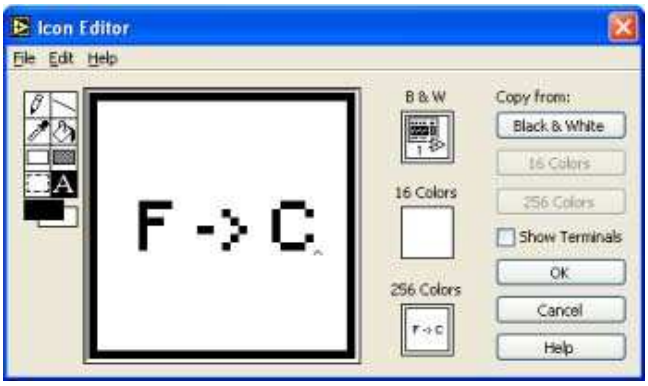

FIGURA 43 - ICON DE UM VI QUANDO USADO EM OUTRO VI'S

4.4.1. Método de programação

Tal como anteriormente mencionado, o *Labview* é uma linguagem de programação gráfica que faz uso de blocos de funções denominados de VI's. Desta maneira, o programador apenas tem que seleccionar os blocos, ou sub VI's, que pretende para a sua aplicação e interliga-los entre si através de linhas. À semelhança de outras linguagens, as sub-rotinas ou no caso do *Labview*, os vários sub-VI's constituintes de um programa, possuem variáveis de entrada e de saída, sendo estas ligadas entre si da maneira que o programador achar mais conveniente.

A execução de um programa em *Labview* é baseada em fluxo de dados, ao contrário do que acontece com outras linguagens de programação, o que significa que a chegada de dados a um determinado "nó" determina a ordem da execução de um determinado VI. Um programador ao construir uma aplicação vai introduzir vários VI's. Desta maneira pode ordenar esses mesmos VI's de maneira a ter o fluxo de dados pretendido. Outro aspecto interessante deste método de programação é o facto de este poder executar múltiplos processos em paralelo.

Uma outra característica bastante interessante nesta linguagem de programação é o facto de muitos dos seus VI's serem polimórficos, ou seja, são capazes de aceitar vários tipos de dados à entrada. Este facto evidencia bastante a versatilidade e a flexibilidade de que esta linguagem de programação dispõe.

Uma das grandes vantagens desta linguagem de programação é a construção de uma interface gráfica de forma relativamente simples e rápida. O *Front Pannel* disponibiliza um conjunto de controlos de vários tipos, tais como, botões interruptores, controlos numéricos ou controlos tipo slide. Apresenta também um leque bastante alargado de indicadores, tais como, tabelas, gráficos e leds. Todos estes indicadores e controlos têm a vantagem de poder ser alterados para estes apresentarem a informação pretendida da maneira que se achar mais conveniente e prática.

Outra das grandes vantagens que o *Labview* apresenta é a existência vários pacotes extra que disponibilizam funções para o controlo do mais variado tipo de dispositivos. Existe por exemplo um pacote para programação de FPGA's, um pacote para controlo de um dispositivo em Tempo Real ou ainda um pacote específico para controlo e simulação. Nesta dissertação irão ser usados alguns destes pacotes, o *FPGA module*, o *Real Time Module* e o *SoftMotiom module*.

4.5. Controlador de tempo real *Compact* Crio

O controlador que se pretendia usar nesta dissertação é o *Compact* Crio 9002 da *National Instruments*. Uma imagem deste controlador pode ser vista na Figura 44.

FIGURA 44 - CONTROLADOR CRIO 9002 A UTILIZAR NA DISSERTAÇÃO

O *Compact* Crio possui um processador *Pentium* de 195 MHz, possui também 32 MB de memória dinâmica e ainda mais 64 MB de memória não volátil, para armazenamento de ficheiros. No que toca capacidade de comunicação com o exterior,

este controlador possui uma porta *Ethernet* 10/100 Mbits por segundo, e ainda uma porta de comunicações RS-232. Além da porta *Ethernet*, este controlador tem já incorporado um servidor HTTP e outro de FTP. Finalmente, salienta-se que a tensão de alimentação deste dispositivo varia entre 6V e 35V DC.

Embora o Compact Crio seja apresentado como uma peça única, este é constituído por três partes distintas: o controlador já apresentado, o chassis reconfigurável e as cartas de aquisição de sinal.

Este *Compact* Crio e os seus módulos constituintes não foram usados para se realizarem alguns testes de *interface*, com a carta de aquisição de sinais descrita neste capítulo da dissertação. Recorrendo ao *FPGA module* programou-se uma aplicação cujo objectivo foi ler e escrever nas estradas e saídas da carta de aquisição de sinais. Através da programação desta aplicação conseguiu-se um entendimento mais assertivo e concreto do método de programação do *Compact* Crio, para que quando os servo drives da série MSD chegassem existisse o menor número de dúvidas possíveis em relação ao capítulo da programação do *Compact* Crio.

4.5.1. Chassis reconfigurável

O chassis reconfigurável é um dos componentes vitais que constituem o Crio, uma vez que no seu interior existe uma FPGA reconfigurável. A FPGA contém uma entrada individual para cada módulo de entradas e saídas, sendo programada com funções bastante simples como a de leitura e escrita de valores nos diferentes módulos. Esta mesma FPGA está ligada ao controlador de tempo real, através de um barramento PCI, podendo existir desta maneira uma troca de dados entre controlador e FPGA. Além da troca de dados entre controlador e FPGA, esta ainda pode gerar interrupções de modo a que ser sincronizada com o controlador de tempo real. Na Figura 45 é apresentada uma imagem deste componente.

FIGURA 45 - CHASSIS RECONFIGURÁVEL UTILIZADOR PELO COMPACT CRIO

4.5.2. Módulos de I/O ou cartas de aquisição de sinais

Este tipo de dispositivos é responsável por gerar os sinais previamente programados. Existem variadíssimos tipos destes módulos, por exemplo, cartas que só disponibilizam saídas analógicas, outros que disponibilizam entradas digitais, e outros que disponibilizam entradas e saídas analógicas ou digitais. É neste componente que se encontra toda a electrónica de acondicionamento de sinal e circuitos de conversão de digital para analógico, DAC, ou de analógico para digital, ADC. Na Figura 46 encontra-se, a título de exemplo, a carta de aquisição de sinais NI 9401. Esta carta possui oito canais independentes, podendo estes ser configurados para entradas ou saídas digitais.

FIGURA 46 - CARTAS E AQUISIÇÃO DE SINAIS NI 9401

Capítulo 5

5. Estrutura do sistema

Depois de no capítulo anterior se ter feito uma apresentação de todo o trabalho que foi efectuado para o manipulador disponibilizado pela empresa onde a dissertação decorreu, neste capítulo será apresentado o simulador que foi construído para que se tivesse um melhor conhecimento de todos os métodos de programação de uma aplicação deste tipo em *Labview*. O objectivo principal do desenvolvimento deste simulador foi o de permitir que os trabalhos de desenvolvimento da interface de controlo do manipulador real pudessem decorrer normalmente, evitando que ficassem demasiadamente dependentes da chegada de equipamento de actuação dos motores do manipulador. Uma vez que detectara que, no decorrer dos trabalhos, havia equipamento em falta (como os drives dos motores) que teria que ser adquirido, sem o qual não seria possível colocar o manipulador em funcionamento, decidi, então, recorrer ao desenvolvimento de um manipulador virtual que permitisse implementar e testar o software de controlo a criar. Desta maneira ao longo deste capítulo serão apresentadas todas as ferramentas utilizadas na construção do simulador.

Convém ainda salientar que muito embora a construção de um simulador não fosse o objectivo inicial desta dissertação, este desempenhou um papel fundamental no melhor entendimento de toda a problemática inerente aos manipuladores, bem como ao conhecimento da linguagem de programação *Labview*.

5.1. Objectivo do simulador

Tal como já foi referido anteriormente, a construção deste simulador não foi o objectivo inicial desta dissertação, tendo este sido desenvolvido para colmatar a falta de material colocado à minha disposição. Desta maneira vi-me na necessidade de procurar alternativas para o desenvolvimento da dissertação.

O simulador desenvolvido usa um manipulador PUMA 760 e tem por base a linguagem de programação *Labview*. Contudo tem a possibilidade de ligação a outras aplicações de software externas a este, no caso, *Matlab* e *Roboworks*. O *Matlab* é usado para implementação de funções onde sejam executados exclusivamente cálculos matemáticos. O *Roboworks* irá permitir a visualização do movimento do manipulador usado para a simulação. Finalmente, o *Labview* irá fazer toda a parte de interface, implementação da malha de controlo, ligação a dispositivos externos e a aplicações variadas.

No simulador desenvolvido, a partir de uma interface gráfica, simples e amigável, o utilizador pode testar a cinemática directa e inversa do manipulador, verificar o comportamento do sistema em malha fechada e programar um trajecto que o manipulador deva executar. O trajecto que o manipulador deve efectuar pode ser introduzido via computador ou colocando o manipulador em modo manual, e levando o mesmo até aos pontos pretendidos. Em modo manual o manipulador recebe comando do teclado do próprio computador, bem como de um dispositivo externo ao computador, como por exemplo um *joystick*.

Uma vez que se pretendia que a aplicação fosse o mais aproximada possível do caso real, foi introduzido um bloco capaz de fornecer dinâmica a todo o sistema. Este bloco é extremamente importante pois permite não só uma simulação bastante mais real, bem como permite também ao utilizador o dimensionamento e teste de compensadores do sistema. Com esta última característica o utilizador pode dimensionar um compensador para que o sistema apresente uma determinada resposta, e visualizar a mesma numa interface gráfica, podendo também visualizar ao longo do tempo parâmetros como a velocidade angular e a posição angular do veio dos vários motores do manipulador.

Depois de se ter feito uma breve e sucinta apresentação de todas as potencialidades do simulador, de seguida será feita a descrição individual dos vários blocos que constituem esta aplicação.

5.2. RoboWorks

O *RoboWorks* é uma ferramenta que permite a criação, animação e simulação 3D de múltiplos cenários ou mesmo de dispositivos físicos. Este software, criado pela empresa Newtonium, é ideal para a modelação e simulação de sistemas mecânicos. Na Figura 47 encontram-se dois exemplos de animação. Esta ferramenta é bastante utilizada ao nível didáctico uma vez que é de simples programação e possui já uma interface de comunicações com outros ambientes de programação como por exemplo o *Labview*, *Matlab* ou Visual Basic.

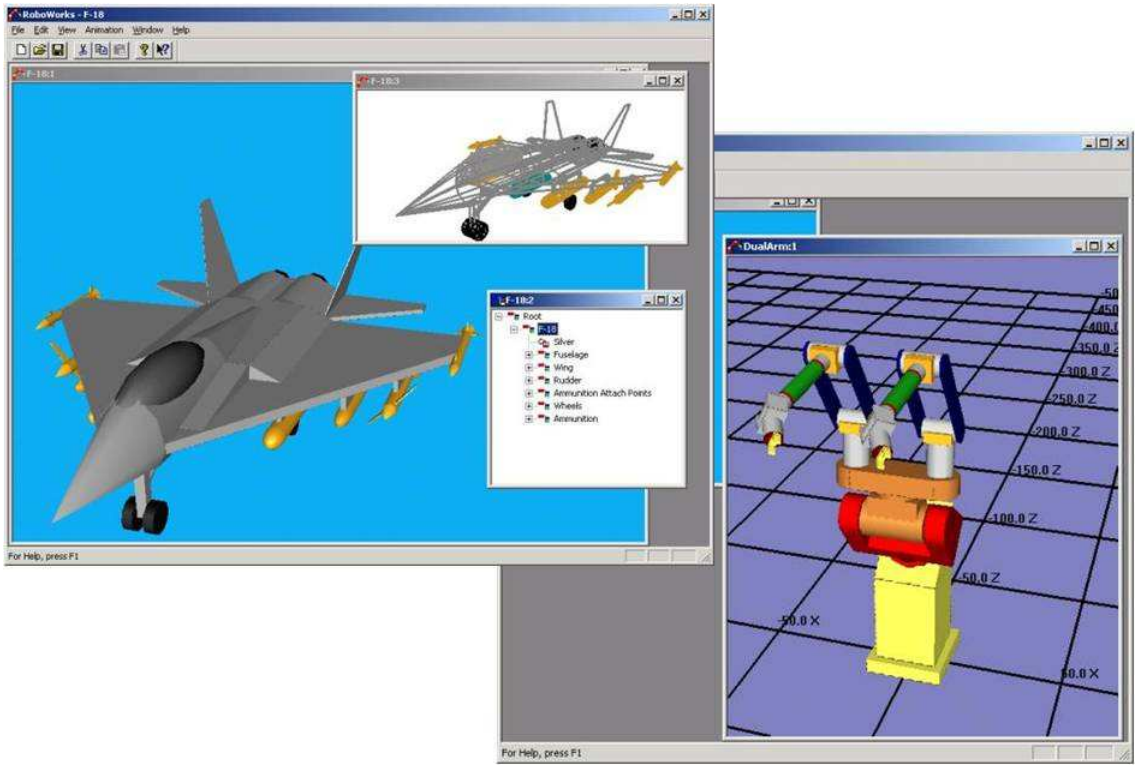

FIGURA 47 - MODELAÇÃO DE UM AVIÃO F-18 E ROBOT DE DOIS BRAÇOS

Este programa, permite também o teste de todo o bloco de comunicações entre controlador e dispositivo, já que o *RoboWorks* disponibiliza uma plataforma de comunicações TCP/IP de tempo real. Por último, e não menos importante, falta apenas referir o facto de esta plataforma já disponibilizar um vasto leque de exemplos e aplicações demonstrativas no seu site, tal como é facilmente perceptível pela Figura 47. [17]

Tal como já foi adiantado, o *RoboWorks* possui não só, toda uma interface com plataformas de programação externas, como também uma simples e intuitiva interface com o utilizador. Estes dois factores fazem deste software a ferramenta ideal para a simulação e teste de todo o software da dissertação.

5.3.Constituição do simulador

O simulador que foi programado foi dimensionado para se aproximar o mais possível da realidade, desta maneira seguiu a estrutura típica que uma aplicação de controlo de um manipulador normalmente tem, ou seja, possui três malhas paralelas. A malha de interface com o utilizador, a malha geradora de trajectórias e a malha de controlo da trajectória. Embora as três malhas corram de forma concorrente, estas possuem diferente periodicidade. Por exemplo, quando o manipulador PUMA 760 está em movimento, a malha de controlo é executada com maior frequência que as restantes. Para que se conheça em maior detalhe as várias malhas do simulador, de seguida será feita uma apresentação sobre o papel que cada uma desempenha. Na Figura 48 encontra-se um esquema que pretende ilustrar o funcionamento das três malhas.

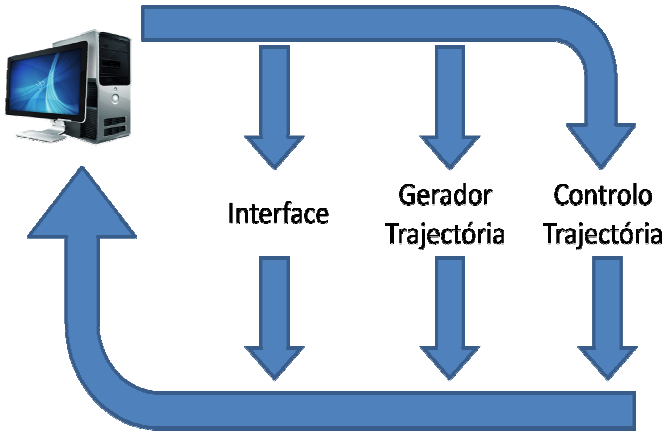

FIGURA 48 - TRÊS MALHA DE EXECUÇÃO DO SIMULADOR

A malha de interface do simulador é, conforme já foi adiantado e tal como o próprio nome sugere, a malha através da qual o utilizador envia comandos e visualiza dados sobre o processo do manipulador. Os comandos enviados para o manipulador podem ser, por exemplo, de início de um determinado trajecto, e de paragem do

mesmo. No que toca à visualização de informação, essa pode ser de várias ordens, desde tendências de velocidade e posição angular de um determinado motor, à resposta impulsional de um determinado motor com um determinado compensador. No simulador, esta malha é implementada em dois ciclos *while*, que correm continuamente. O primeiro é uma estrutura de eventos, que executa um determinado trecho de código cada vez que um dos eventos ocorre. No caso deste simulador, os eventos são a mudança de estado dos botões do interface. O segundo *while* desta malha controla as várias fases de execução do simulador. Estes dois ciclos serão explicados em maior detalhe numa fase mais adiantada da dissertação.

A malha geradora de trajectórias é responsável por gerar todo o perfil de movimento de um determinado trajecto. Esse perfil irá obedecer às constantes definidas para a velocidade, aceleração e desaceleração pretendida para o movimento. Estas três constantes terão que ser obrigatoriamente definidas antes do início do movimento. Esta malha será responsável por gerar pontos intermédios entre o ponto de origem e o de destino. Esses pontos são gerados com uma determinada periodicidade e têm em conta o perfil de movimento definido. Uma vez que podem ser implementadas diversos tipos de malha de controlo, esta malha gera não só pontos intermédios, como também valores de velocidade aceleração, e os quatro coeficientes de uma *spline* cúbica para esse ponto. No simulador a malha de controlo é implementada num ciclo *while* de tempo real que ocorre com uma determinada periodicidade. O funcionamento deste ciclo será explicado numa fase mais adiantada da dissertação.

A terceira, e última, malha de controlo é a malha que durante a execução de uma trajectória controla o movimento do manipulador, fazendo com que este siga a trajectória pretendida. Quando o manipulador está em movimento esta malha é a que terá maior frequência de execução de todas as três malhas do simulador. Uma vez que esta malha é vital para que a trajectória seguida pelo manipulador seja correcta, esta malha é tipicamente implementada num controlador de tempo real. Se todo o material tivesse chegado em tempo útil, esta seria implementada na *FPGA* do controlador *CompactRio*. Neste simulador foi implementada uma malha de controlo da

posição do manipulador, recorrendo para tal a um bloco que simula o comportamento de um motor *DC* sem escovas, do mesmo tipo que se pretendia usar no manipulador real. No simulador desenvolvido, esta malha de controlo é na realidade um conjunto de três malhas de controlo, um para cada uma das três primeiras juntas do manipulador. Toda esta malha, bem como todos os blocos que a constituem serão explicados numa fase mais adiantada da dissertação.

Em suma estas três malhas são como que a espinha dorsal do manipulador, desempenhando cada uma delas um papel diferente e complementar. O mau funcionamento de uma das malhas terá obrigatoriamente repercussões nas restantes.

5.4.Interface simulador

Quando a interface principal do simulador, que foi desenvolvida em *Labview*, é inicializada, a aplicação gera um menu onde o utilizador tem diversas opções à escolha. Uma imagem desse mesmo menu é apresentado na Figura 49. Nesta figura é possível ver a existência de três blocos: tipos de movimento, informação referente ao *Roboworks* e o *tuning* do controlador.

Nas opções do primeiro bloco referido, tipos de movimento, o utilizador pode escolher vários tipos de movimento para o manipulador. Este bloco pode ser dividido em dois sub-blocos. O primeiro sub-bloco é constituído pelas opções: cinemática directa, cinemática inversa e comando manual. O segundo sub-bloco é constituído pelas opções: movimento simples e Trajecto. A diferença entre estes dois sub-blocos é que, ao contrário do que acontece com o segundo sub-bloco, as malhas de controlo e geradora de trajectórias não são executadas. Isto significa que o manipulador não possui qualquer tipo de dinâmica.

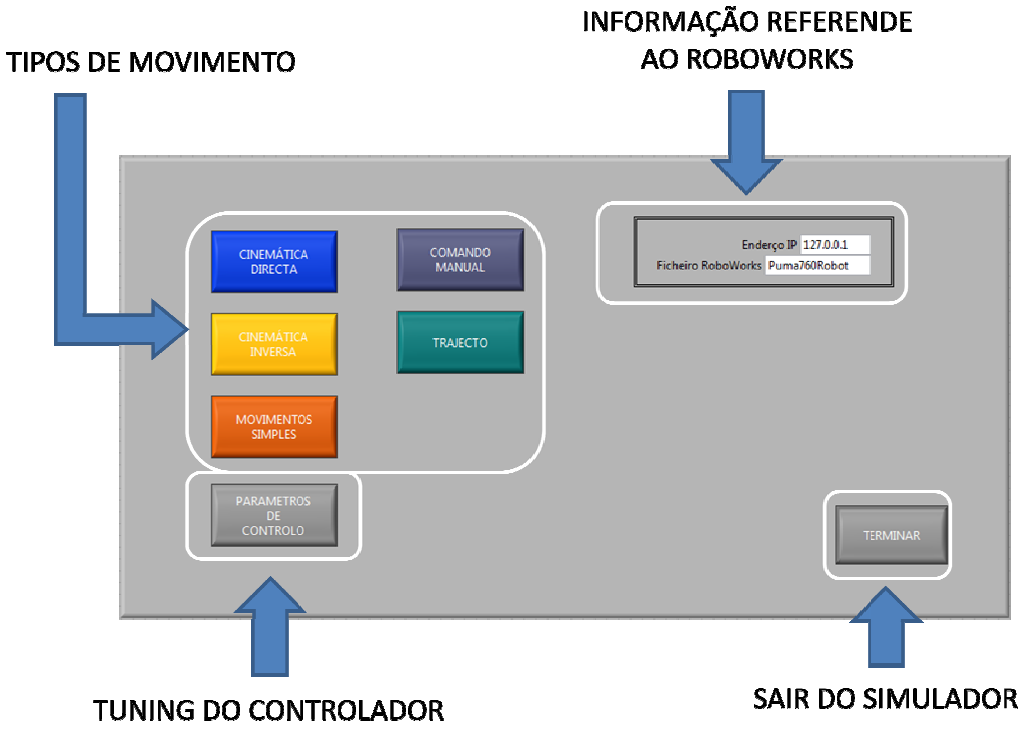

FIGURA 49 - MENU INICIAL DO SIMULADOR

No segundo bloco que surge na interface do simulador, informação referente ao *Roboworks*, é visível toda a informação necessária para que todos os dados produzidos pela aplicação de *Labview* seja enviada correctamente para o *Roboworks*. Uma vez que o ambiente de simulação do manipulador PUMA 760 pode estar a correr num outro computador ligado em rede ao que está a correr a aplicação de interface desenvolvida em *Labview*, é necessário introduzir o IP da máquina de destino, bem como o nome do processo que este toma na máquina de destino.

No terceiro, e último bloco, *tuning* do compensador, o utilizador pode proceder à escolha do compensador a utilizar na malha de controlo, bem como proceder a ajustes que achar necessários. Através deste bloco o utilizador pode antever a resposta do sistema em malha fechada.

5.4.1. Cinemática directa

Quando o utilizador prime a tecla da Cinemática Directa irá surgir um novo menu ao utilizador, tal como apresentado na Figura 50.

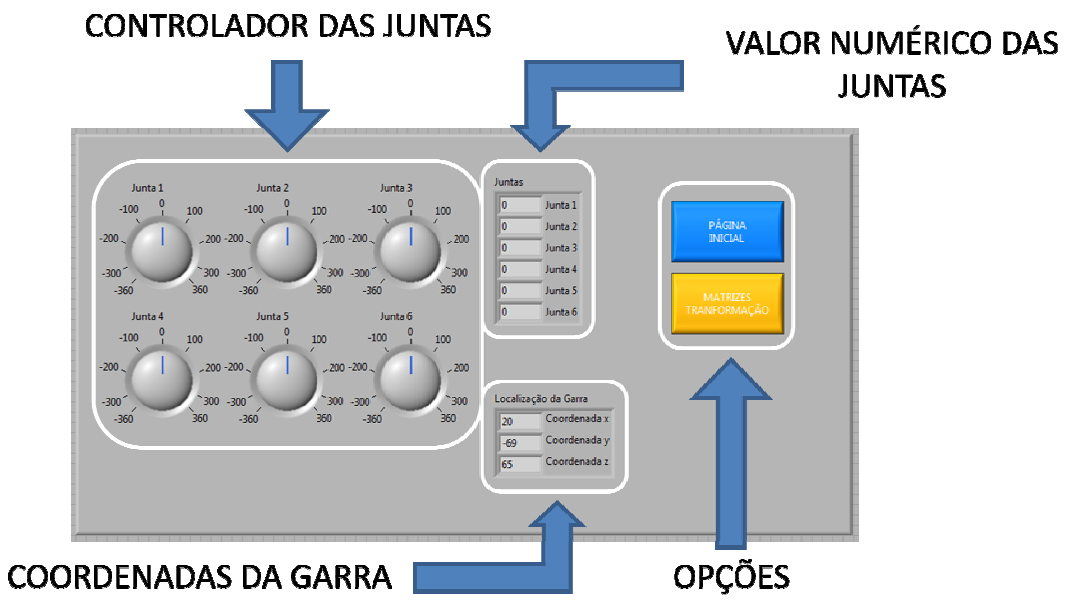

FIGURA 50 - INTERFACE COM O SIMULADOR DEPOIS DE PRESSIONADO O BOTÃO 'CINEMÁTICA DIRECTA' NO MENU INICIAL

Este novo menu é constituído por um conjunto de controlos e indicadores que podem ser divididos em quatro grupos: controladores das juntas do manipulador, indicadores numéricos das juntas, localização da garra e opções à disposição.

No primeiro grupo de controlos o utilizador pode alterar o valor numérico de cada junta do manipulador, bastando para tal rodar o controlo da junta pretendida.

O segundo grupo serve para que o utilizador possa ver em mais detalhe qual o valor numérico de cada junta. O terceiro grupo possui três indicadores, indicando cada um a localização da garra segundo um eixo de coordenadas do referencial da base do manipulador.

Finalmente, o quarto grupo possui dois botões: Página Inicial ou Matrizes de Transformação. Quando o primeiro botão é pressionado a aplicação retorna ao menu inicial. Quando o segundo botão é pressionado o utilizador salta de imediato para outro menu, apresentado na Figura 51, onde pode visualizar as várias matrizes de transformação definidas para o manipulador.

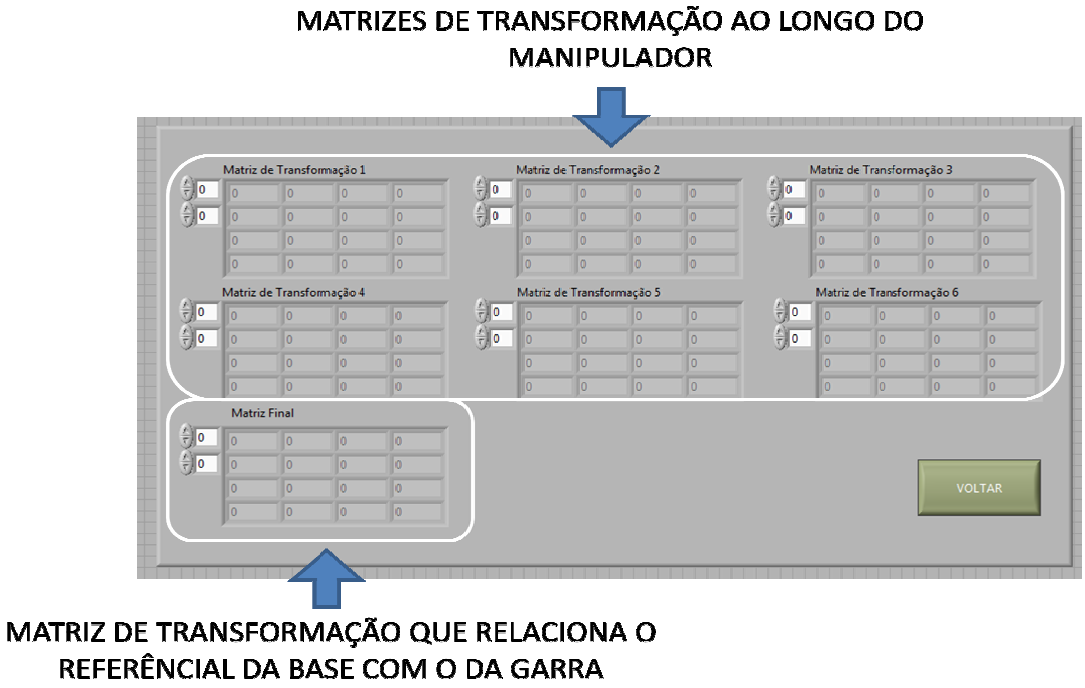

FIGURA 51 - **MENU DE VISUALIZAÇÃO DE MATRIZES DO MANIPULADOR**

5.4.2. Cinemática inversa

Quando o utilizador está no menu inicial e prime a tecla Cinemática Inversa, irá surgir um novo menu ao utilizador, apresentado na Figura 52.

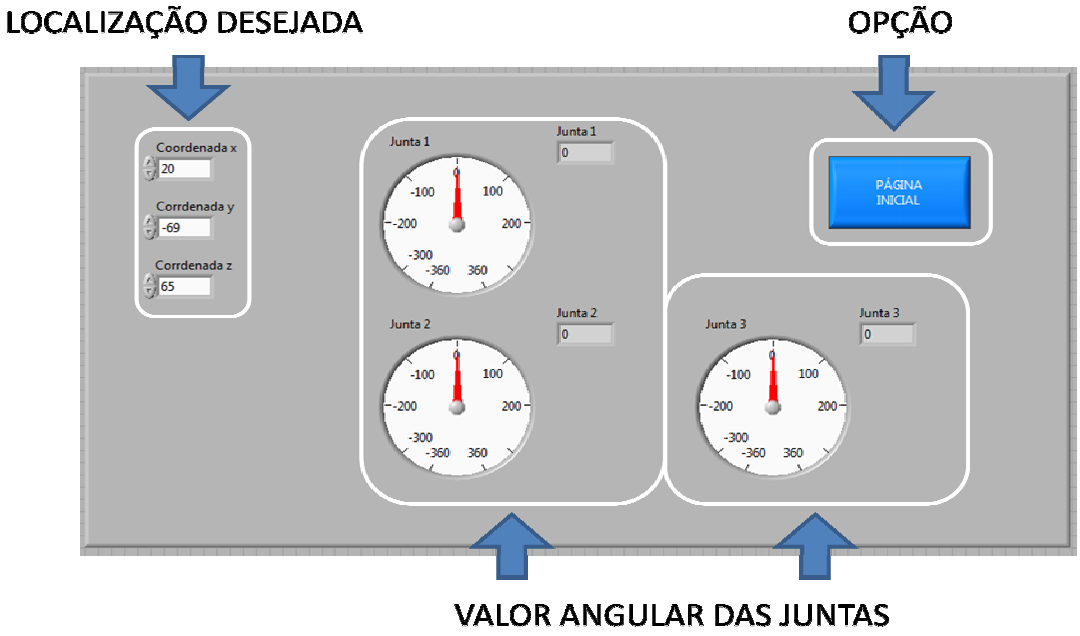

FIGURA 52 - INTERFACE DO SIMULADOR DEPOIS DE PRESSIONADO O BOTÃO 'CINEMÁTICA INERSA'

Este novo menu irá ser constituído por um conjunto de controlos e indicadores numéricos onde o utilizador pode visualizar ou alterar a localização da garra do manipulador. Este menu pode ser divido em três grupos: controladores da localização da garra, valor angular das juntas e por último uma opção.

O primeiro grupo é constituído por três controladores numéricos, onde o utilizador pode introduzir o valor pretendido para cada coordenada. Esta localização da garra será sempre referente ao sistema de coordenadas definido na base do manipulador.

O segundo grupo é formado por seis indicadores, três numéricos e outros três gráficos. Nestes seis indicadores o utilizador pode ver a posição angular das três juntas do manipulador. A posição angular das juntas é calculada através das equações da cinemática inversa do manipulador PUMA 760. Tais equações foram retiradas do livro [9] e serão apresentadas numa fase mais adiantada desta dissertação.

Finalmente, o terceiro e último grupo é constituído por apenas um botão, que o utilizador pode pressionar se desejar voltar ao menu inicial.

5.4.3. Movimento simples

Quando o utilizador está no menu inicial e prime a tecla Cinemática Inversa, irá surgir um novo menu ao utilizador, descrito na Figura 53.

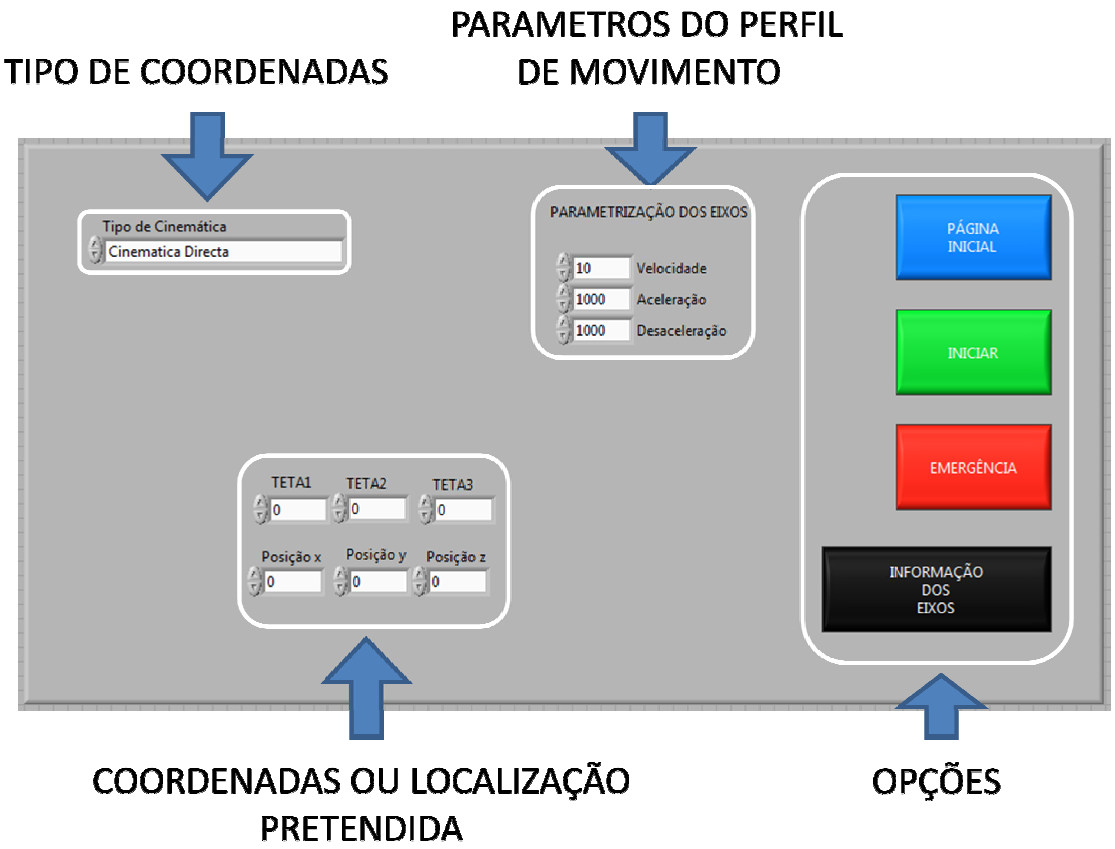

FIGURA 53 - INTERFACE COM O SIMULADOR DEPOIS DE PRESSIONADO O BOTÃO 'MOVIMENTO SIMPLES' NO MENU INICIAL

Neste novo menu o utilizador pode introduzir as coordenadas da localização pretendida para a garra e verificar um movimento contínuo do manipulador. O manipulador ao descrever um movimento contínuo irá seguir um determinado perfil de movimento definido *a priori* através dos parâmetros do perfil de movimento definidas pelo utilizador na interface. Os três parâmetros definidos pelo utilizador irão ser usado no ciclo gerador de trajectórias. Tal como se observa na Figura 53, este menu pode ser dividido em quatro grupos: tipo de coordenadas, parâmetros do perfil de movimento, coordenadas ou localização pretendida e opções.

O primeiro grupo é constituído por um selector, onde o utilizador pode escolher o tipo de coordenadas que irá introduzir para a localização da garra do manipulador. Este controlador oferece duas possibilidades: cinemática directa e cinemática inversa. Se utilizador escolher a primeira possibilidade, então irá introduzir dados no terceiro grupo, concretamente a posição angular da localização pretendida

para a garra. Se, pelo contrário, escolher a segunda possibilidade, então irá introduzir as coordenadas cartesianas pretendidas nas caixas do terceiro grupo.

O segundo grupo é constituído por três controlos numéricos: a velocidade, a aceleração e a desaceleração. O perfil de movimento definido para o manipulador é o trapezoidal, assim estes três parâmetros irão definir a inclinação da rampa de aceleração e desaceleração do manipulador de uma junta até esta atingir a velocidade pretendida.

O terceiro grupo é constituído por seis controlos: TETA1, TETA2, TETA3, Posição x, Posição y e Posição z. Os três primeiros controlos só serão necessários se no primeiro grupo o tipo de coordenadas escolhido for a cinemática inversa. No primeiro controlador, TETA1, o utilizador introduz a posição angular pretendida para o motor da base do manipulador, sendo o segundo e terceiro controlador referentes ao segundo e terceiro motores do PUMA 760. Os quarto, quinto e sexto controlos só serão necessários se no primeiro grupo o tipo de coordenadas definido for a cinemática inversa. O quarto controlador, Posição X, define a localização pretendida para a garra do manipulador segundo o eixo xx do referencial da base do manipulador. Os quinto e sexto controlos definem a localização da garra segundo os eixos yy e zz, respectivamente.

Finalmente, o quarto grupo é constituído por um conjunto de quatro botões: Página Inicial, Iniciar, Emergência, Informações dos eixos. O primeiro botão, à semelhança das interfaces anteriormente apresentadas, é utilizado se se pretender voltar ao menu raiz do simulador. O segundo botão é utilizado para iniciar o movimento para a localização definida pelo utilizador. O terceiro botão é utilizado no caso de o utilizador pretender parar o manipulador durante o seu trajecto. O quarto e último botão é utilizado caso o utilizador pretenda ver informação inerente à trajectória efectua. Quando este botão é pressionado irá o menu da Figura 54.

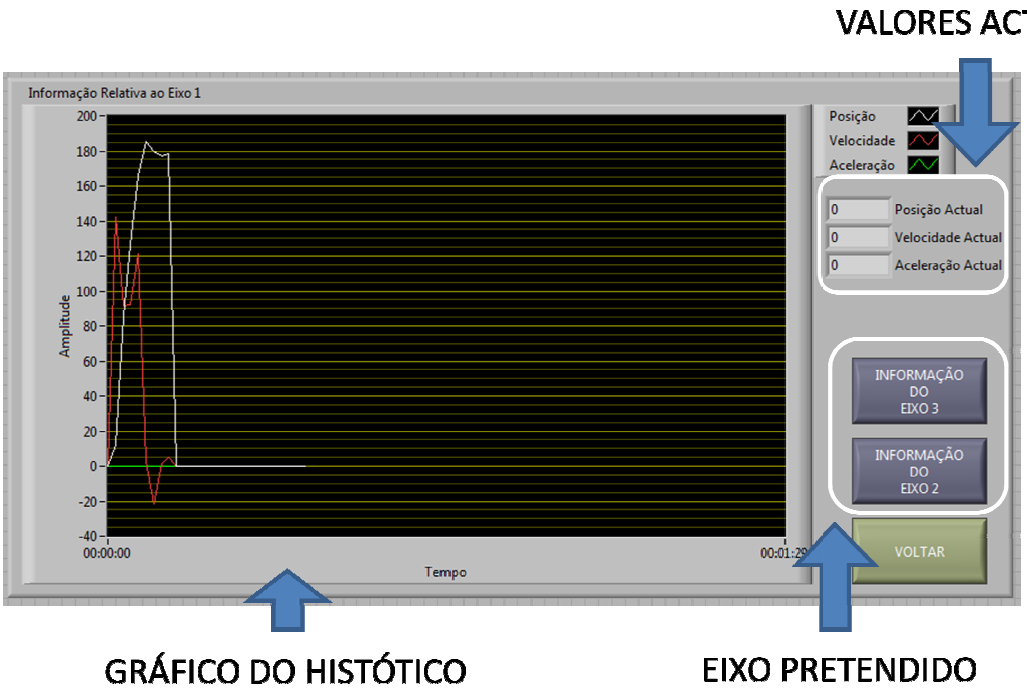

FIGURA 54 - INTERFACE COM INFORMAÇÃO DE MOVIMENO DE CADA EIXO

A Figura 54 apresenta a informação relativa a um movimento efectuado pela primeira junta. Nesta figura o utilizador tem à sua disposição dois grupos, um de dois botões e outro de três indicadores numéricos, sendo visível também um gráfico da variação da posição, velocidade e aceleração ao longo do tempo. O grupo de indicadores numéricos é constituído por três parâmetros: posição, velocidade e aceleração actual. Nestes três indicadores o utilizador pode visualizar com maior detalhe os valores actuais dos três parâmetros. No grupo formado pelos dois botões, o utilizador pode escolher o eixo que pretende visualizar. Se um destes dois botões for pressionado então irá surgir outro menu idêntico ao da Figura 54.

5.4.4. Parâmetros de controlo

Quando o utilizador está no menu inicial e prime a tecla Parâmetros de Controlo irá surgir um novo menu ao utilizador, como apresentado na Figura 55.

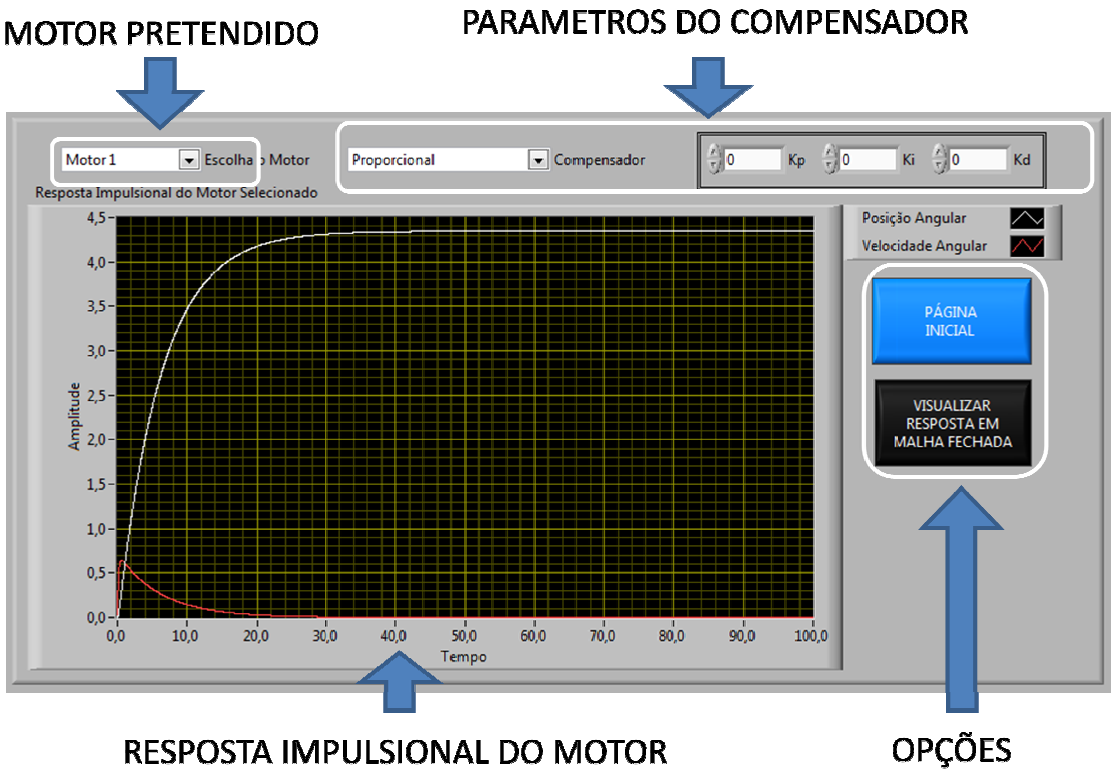

FIGURA 55 - INTERFACE COM O SIMULADOR DEPOIS DE PRESSIONADO O 'BOTÃO PARÂMETROS' DE CONTROLO NO MENU INICIAL

Neste novo menu é possível dimensionar um compensador que melhor se adapte ao sistema. Como cada motor tem uma malha de controlo individual, o utilizador pode definir um compensador para cada um dos três motores utilizados no simulador. Tal como se pode ver na Figura 55 o utilizador pode seleccionar o motor que pretende compensar através de um controlo localizado no canto superior esquerdo do menu. O utilizador pode também seleccionar o tipo de compensador que pretende utilizar, bem como alterar os seus parâmetros. Tal pode ser feito se se utilizar o conjunto de controlos definido na Figura 55 como Parâmetros de Controlo. No gráfico presente na Figura 55 pode-se verificar a resposta impulsional da função de transferência que relaciona a posição angular do motor com a tensão que é colocada aos seus terminais, bem como a resposta impulsional da função de transferência que relaciona a velocidade angular com a sua tensão de alimentação. Existem também dois botões que o utilizador pode premir: Página Inicial e Visualizar Resposta em Malha Fechada. À semelhança do que já foi dito em pontos anteriores desta dissertação, este botão serve para o utilizador voltar ao menu raiz do simulador. Quando o segundo

botão é premido irá aparecer um novo menu onde o utilizador pode verificar a resposta impulsional de todo o sistema, motor e compensador, em malha fechada. Esse novo menu será em tudo idêntico ao que está na Figura 56.

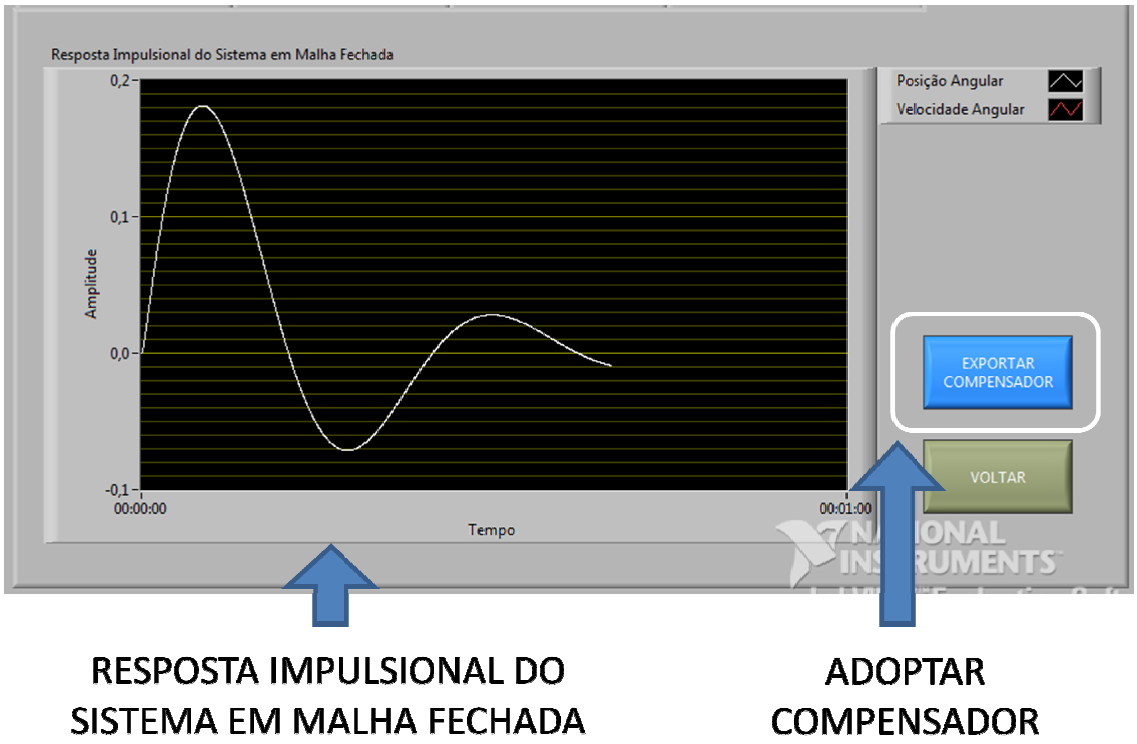

FIGURA 56 – INTEFACE DA RESOSTA IMPULSIONAL DO SISTEMA EM MALHA FECHADA

Neste novo menu o utilizador pode verificar a resposta impulsional da função de transferência de todo o sistema. Para melhor se entender a resposta impulsional que é colocada no gráfico da Figura 56 pode observar-se o diagrama presente na Figura 57.

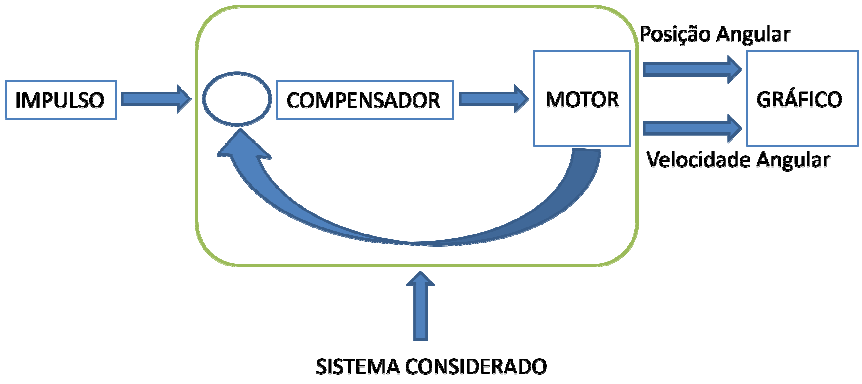

FIGURA 57 - DIAGRAMA DE BLOCOS UTILIZADO PARA CALCULAR A RESPOSTA IMPULSIONAL DO SISTEMA EM MALHA FECHADA

Para terminar a análise do menu presente na Figura 56 falta explicar a função do botão Exportar Compensador. Quando este botão é premido, e tal como a Figura 56 transmite, o compensador dimensionado é passado para a raiz do simulador. Ou seja, quando o utilizador efectuar um movimento contínuo com o manipulador, o compensador que será usado na malha de controlo será o que foi dimensionado. Através desta ferramenta o utilizador pode em ante mão dimensionar o comportamento do sistema.

5.4.5. Comando manual

Quando o utilizador está no menu inicial e prime a tecla Comando Manual, irá surgir um novo menu ao utilizador, esse novo menu será em tudo igual ao da Figura 58.

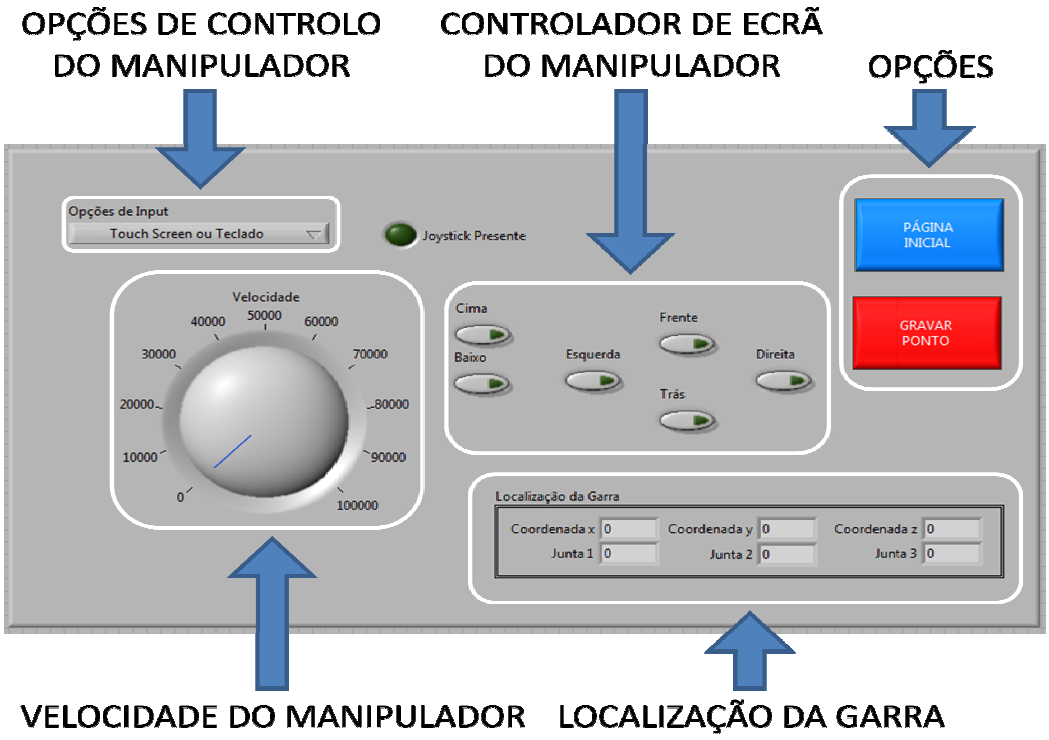

FIGURA 58 - INTERFACE COM O SIMULADOR DEPOIS DE PRESSIONADO O BOTÃO 'COMANDO MANUAL' NO MENU INICIAL

Neste menu o utilizador pode definir um trajecto a executar pelo manipulador. Para definir um trajecto o utilizador tem que movimentar o manipulador através dos

pontos que pretende que este percorra. Para movimentar o manipulador podem-se utilizar três vias: teclado, botões na interface da Figura 58 e ainda um joystick externo ligado por USB ao computador onde a aplicação está ser executada. A escolha do tipo de *input* do manipulador é feita seleccionando o controlador presente no campo superior esquerdo da interface. Quando o utilizador usa o teclado, as teclas de movimentação do manipulador são as quatro setas, a tecla 'c' para um movimento ascendente da garra segundo o eixo yy e ainda a tecla 'b' para um movimento descendente da garra segundo o eixo anterior. Pode ainda utilizar os botões disponibilizados na interface da aplicação. Os controlos que permitem isso estão localizados na parte superior central do menu, isso mesmo é visível na Figura 58. No que toca ao joystick externo ao manipulador, a configuração deste varia consoante o modelo utilizado. O Joystick só será reconhecido caso este esteja ligado antes a iniciação da aplicação.

No canto inferior esquerdo é apresentado um controlador gráfico onde se pode controlar a velocidade de movimentação do manipulador. Este controlador só é necessário caso se tenha definido como opção de input: Toutch Screen ou Teclado.

A localização instantânea do elemento terminal do manipulador pode ser vista no canto inferior direito da interface. Essa localização é apresentada num grupo de seis indicadores numéricos. Os três indicadores localizados na parte superior deste grupo indicam a localização em coordenadas cartesianas. Os três indicadores localizados na parte inferior deste grupo indicam a posição angular de cada um dos motores do manipulador.

O trajecto do manipulador é definido por vários sub-trajectos. Esses trajectos intermédios são definidos por pontos intermédios ou de passagem. Tais pontos podem ser definidos pelo utilizador levando o manipulador até ao ponto pretendido e, de seguida, premindo a tecla de fundo vermelho 'Gravar Ponto', no canto superior esquerdo do interface visível na Figura 58. Quando tal botão é pressionado é passado para um ficheiro de extensão csv o valor da posição angular dos três motores. O ficheiro pode facilmente ser editado através do programa *Microsoft Excel*. Finalmente,

e à semelhança de todos os interfaces que já foram apresentados, quando o botão 'Página Inicial' é premido a aplicação irá voltar ao menu raiz.

5.4.6. Trajecto

Quando o utilizador está no menu inicial e prime a botão 'Trajecto' irá surgir um novo menu ao utilizador, que é apresentado na Figura 59.

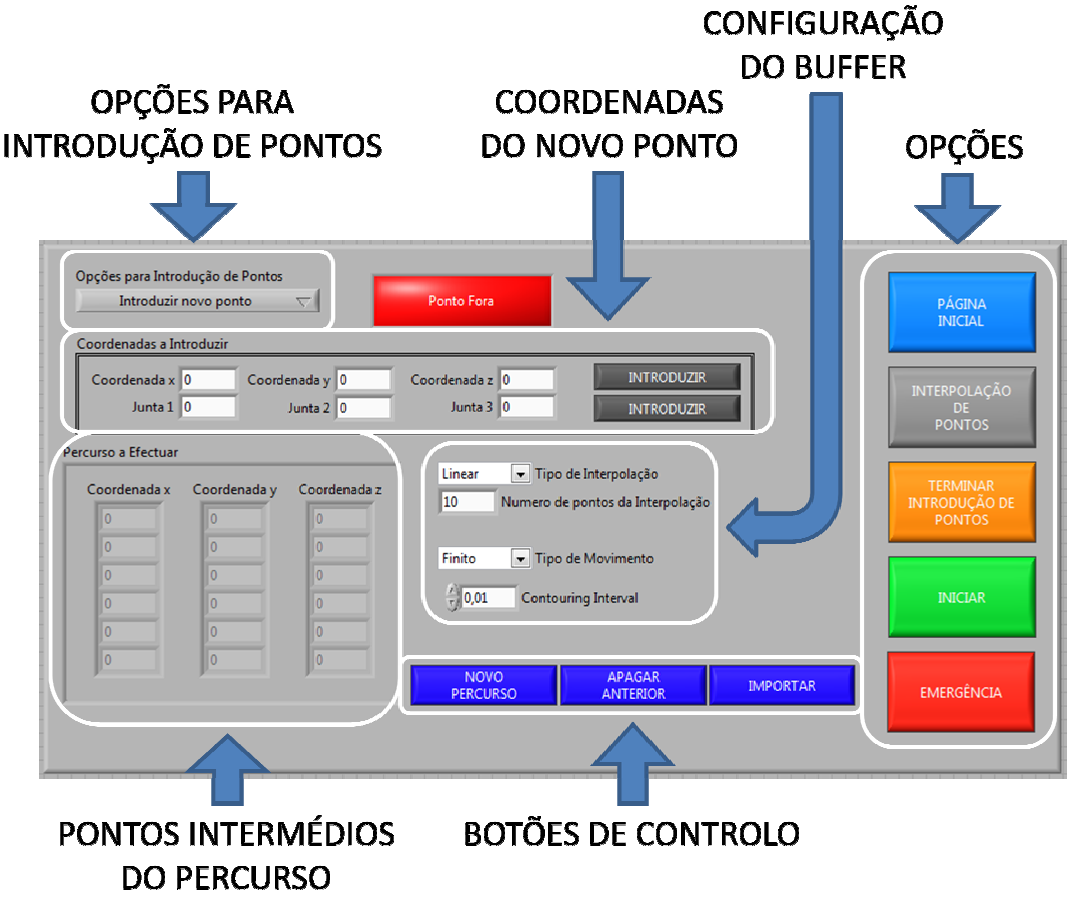

FIGURA 59 - INTERFACE COM O SIMULADOR DEPOIS DE PRESSIONADO O BOTÃO 'TRAJECTO' NO MENU INICIAL

Neste novo menu o utilizador pode definir um novo trajecto a efectuar pelo manipulador, definindo para tal os pontos intermédios de passagem, ou também pode importar o percurso de um ficheiro csv. Para tal é utilizado o controlador localizado no canto superior esquerdo da Figura 59. Este controlador exibe quatro opções: 'Introduzir no ponto','Substituir ponto anterior', 'Importar pontos de ficheiro' e ainda 'Apagar ponto anterior'. Como se pode ver através das opções, este controlo é

utilizado para controlar toda a sequência de introdução de pontos de um trajecto. Todos os pontos introduzidos irão figurar no canto inferior esquerdo do menu apresentado na Figura 59. À semelhança de outros menus, um ponto pode ser definido usando dois métodos: ou através da posição angular dos três motores iniciais do manipulador, ou introduzindo as coordenadas cartesianas que definem o ponto pretendido. Tal pode ser feito através dos seis controlos numéricos localizados na parte centro/esquerda do menu da Figura 59.

Na parte central da interface existe um conjunto de controlos que são referentes à configuração do *buffer* que irá ser gerado. Entenda-se *buffer* gerado como sendo o *array* de pontos gerados para execução do trajecto pretendido. No total existem quatro controlos: 'Tipo de Interpolação', 'Numero de pontos da Interpolação', 'Tipo de Movimento' e 'Countoring Interval'. O primeiro controlo define o tipo de interpolação que é usado para calcular pontos de passagem intermédios entre dois pontos consecutivos da trajectória definida. O segundo controlo pode ser ajustado para definir o número de pontos de passagem intermédios pretendidos entre dois pontos consecutivos da trajectória pretendida. O terceiro controlo é utilizado para definir o tipo de movimento: finito ou contínuo. O movimento finito permite efectuar o trajecto definido uma vez enquanto o movimento continuo efectua o trajecto o número de vezes que o utilizador pretender. O quarto e último controlo pode ser ajustado para aumentar ou diminuir o tempo pretendido entre dois pontos de passagem intermédios calculados pela interpolação.

No canto direito da Figura 59 é possível observar um conjunto de cinco botões: 'Página Inicial', 'Interpolação de Pontos', 'Terminar Introdução de Pontos', 'Iniciar' e 'Emergência'. A função o primeiro botão já foi apresentada em pontos anteriores a este. Quando o segundo botão é premido é feita a interpolação entre pontos consecutivos da trajectória. O terceiro botão deve ser pressionado quando todo o percurso tiver definido e interpolado. O quarto botão é utilizado para dar início ao movimento do manipulador. Finalmente, o sexto e último botão é utilizado para o caso de o utilizador pretender parar o manipulador.

5.5.Estrutura do simulador

Depois de se terem apresentado as três malhas que constituem o simulador e toda a interface, será agora apresentado o fluxo de dados que ocorre durante a execução do simulador. Na Figura 60 encontra-se um diagrama de fluxo de software que mostra como o simulador está organizado.

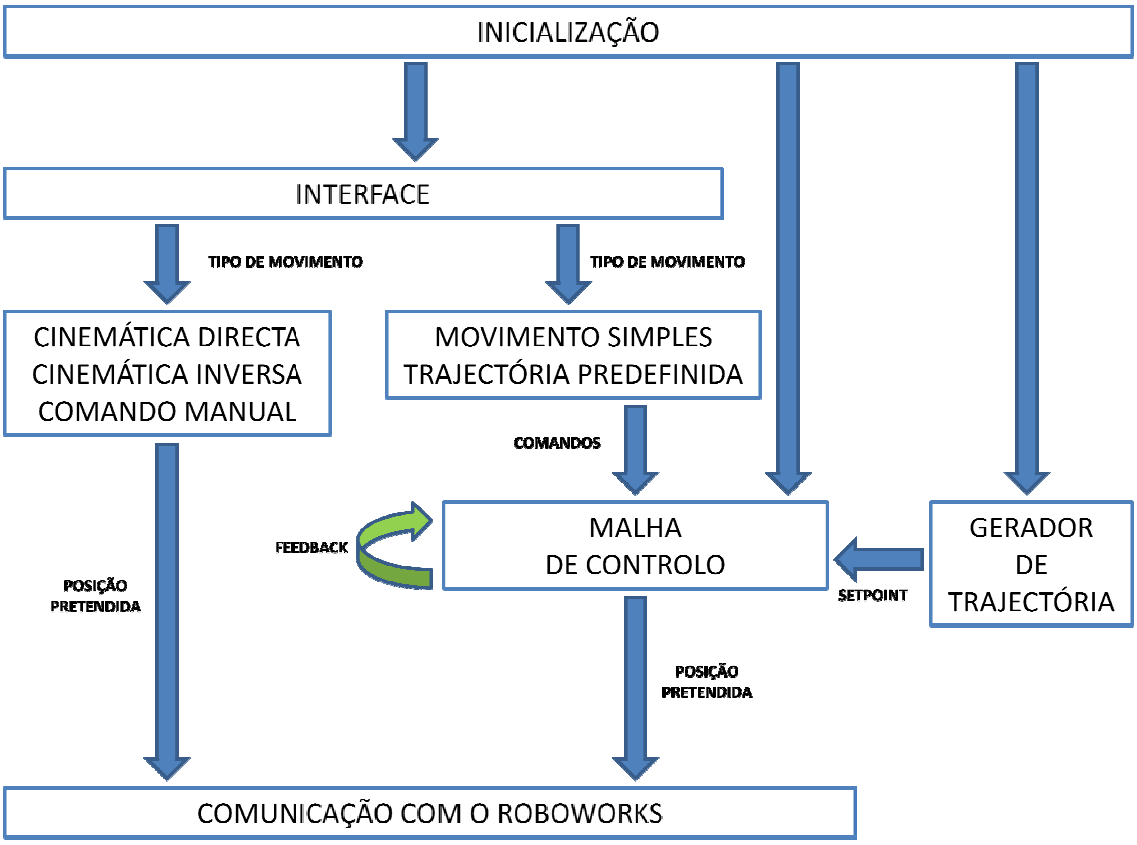

FIGURA 60- DIAGRAMA DE FLUXO DE SOFTWARE

Tal como se pode ver no diagrama anterior, ao se abrir a aplicação é de imediato inicializado o bloco de inicialização. Depois de este ser executado todo o seu conteúdo é a vez de serem executados três outros blocos: interface, malha de controlo e o gerador de trajectórias. Embora estes dois últimos blocos sejam inicializados só serão executados caso o utilizador prima a opção 'Movimento Simples' ou 'Trajectória' no menu raiz da aplicação, presente na Figura 49.

5.5.1. Bloco de inicialização

Tal como já foi referido no ponto anterior, este bloco é o primeiro bloco de toda a aplicação a ser executado. Na aplicação este é representado pelo bloco 'Iniciacao.vi'. Neste bloco serão inicializadas variáveis de três tipos: variáveis de controlo de fluxo da interface, variáveis internas ao gerador de trajectórias e ainda variáveis internas à malha de controlo. A Figura 61 apresenta a imagem do diagrama de blocos de 'Iniciacao.vi'.

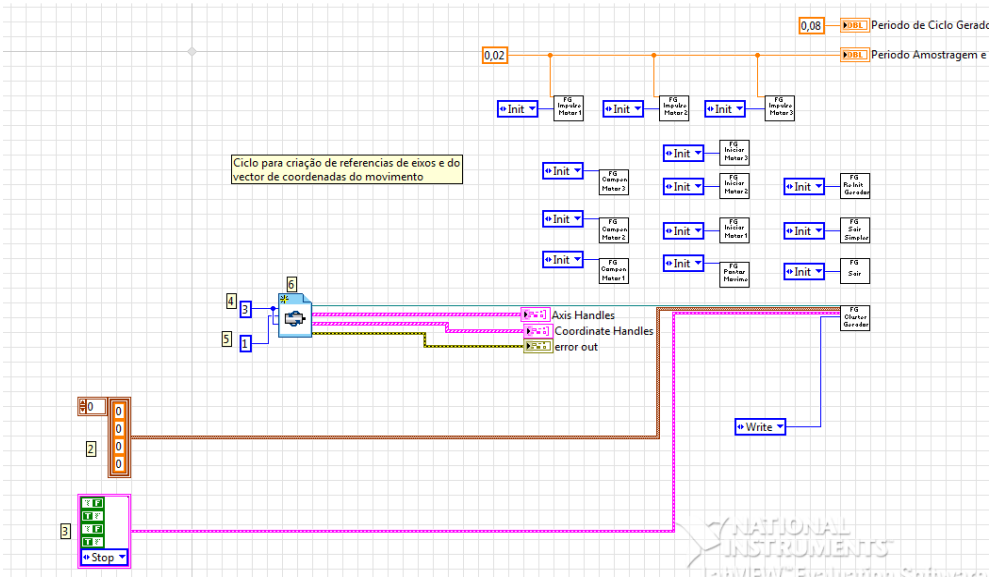

FIGURA 61 - DIAGRAMA DE BLOCOS DO BLOCO 'INICIACAO.VI'

No ponto anterior deste capítulo foi apresentada toda a interface do simulador e, como se observou, esta é um objecto dinâmico. Para controlar o fluxo de dados e a dinâmica da interface são usadas variáveis que têm que comunicar com um sem número de blocos que lhe estão associadas. Desta maneira estas variáveis têm que ser obrigatoriamente inicializadas, não podendo à partida tomar um valor inicialmente aleatório.

No bloco de inicialização são também inicializadas as variáveis necessárias ao bloco gerador de trajectórias. No que concerne ao bloco gerador de trajectórias é necessário definir o número de eixos que irão ser controlados, sendo também necessário definir o número de movimentos coordenados. Quando o manipulador efectua um determinado movimento, este pode ser em um ou vários eixos do mesmo. Desta maneira é necessário que esse mesmo movimento seja coordenado para que o

movimento segundo eixos diferentes não seja descoordenado. Assim os três eixos a controlar no simulador irão formar em conjunto um único movimento coordenado. Ainda em relação ao bloco gerador de trajectórias é necessário inicializar os três *array's*, um por cada eixo, de dados da trajectória, *Trajectory Data*, e ainda inicializar o *array* de estado da execução do movimento, *Status Execution*.

Por último, são inicializadas variáveis necessárias para o controlo da malha de controlo, bem como também são inicializados nove *array's*, três por cada eixo do manipulador. Um *array* de um eixo irá ser inicializado com a resposta impulsional da função de transferência que relaciona a tensão de alimentação com a posição angular do veio do motor com um período de amostragem de 20 ms. O segundo *array* será inicializado com o resposta impulsional da função de transferência que relaciona a tensão de alimentação do motor com a velocidade angular do veio do motor, com o mesmo período de amostragem, ou seja 20 ms. O terceiro e último *array* é inicializado com a resposta impulsional da função de transferência que relaciona a tensão de alimentação com a aceleração angular do veio do motor, novamente amostrada com um período de 20 ms. A resposta impulsional das três funções de transferência é calculada através da função "*impulse*" do *Matlab*. Tal só é possível uma vez que o *Labview* disponibiliza uma ligação a um servidor externo do *Matlab*. As referidas funções de transferência serão apresentadas num ponto mais avançado desta dissertação.

Todas as variáveis inicializadas são passadas para os blocos onde irão ser lidas e escritas durante a execução da aplicação através de *function globals*. O funcionamento destas funções será explicado no ponto seguinte deste capítulo.

5.5.2. Método de comunicação entre blocos do simulador (*Function Global*)

A comunicação entre os vários blocos que constituem a aplicação é vital para um correcto funcionamento desta. Assim, foi necessário dimensionar um bloco que

permitisse efectuar tal missão de forma eficaz. Na Figura 62 encontra-se o diagrama de funcional do bloco projectado.

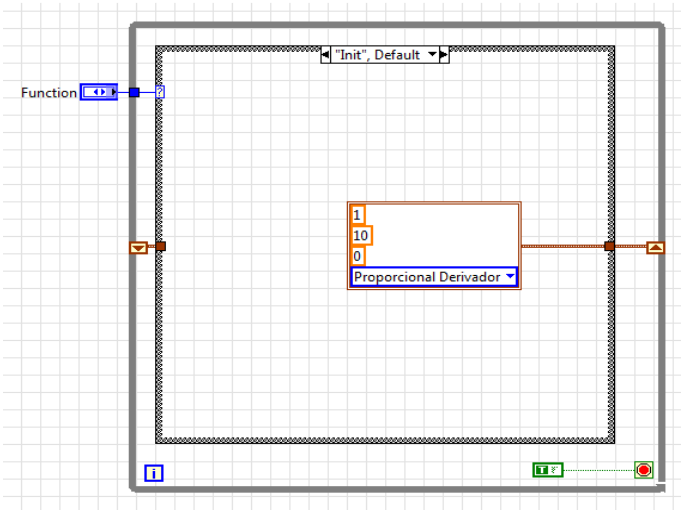

FIGURA 62 - DIAGRAMA FUNCIONAL DO BLOCO DE COMUNICAÇÃO ENTRE VÁRIOS BLOCOS DA APLICAÇÃO

Este bloco, denominado genericamente de *function global*, é contstituido por um ciclo *while* com uma condição de paragem "*stop if is true*". À entrada da condição de paragem foi ligada uma constante booleana, com o valor de *true*. Desta maneira assegura-se que o código contido no interior do ciclo *while* é executado apenas uma vez de cada vez que este bloco é chamado num outro bloco.

Além das variáveis de entrada e saída que se pretendam passar, este bloco terá que conter obrigatoriamente um controlador que defina a fase de operação pretendida. No total existem três fases de operação: inicialização, escrita e leitura. Se se observar com atenção a Figura 62 observa-se que existem inúmeros blocos que têm associados uma constante com o valor 'Init'. Isso significa que todas as variáveis contidas no interior dessa *function global* estão a ser inicializadas. A separação entre fases de operação é feita no interior de ciclo *while* através de um ciclo *case*. No interior do ciclo *case* está contido o código que se pretende executar em cada uma das fases de operação. Por fim existe ainda um *shift register* que assume o papel de memória do bloco. É nesta "memória" que é guardado o último valor passado para o bloco da última vez que este entrou na fase de operação de escrita.

Uma *function global* pode ser usada para passar desde apenas uma variável do tipo booleano, até um *cluster* constituído por várias variáveis.

5.5.3. Constituição dos blocos de interface

No início deste capítulo foi dito que o bom funcionamento da interface com o utilizador é assegurado por dois ciclos *while*. Esses dois ciclos *while*, embora tendo funções diferentes, complementam-se mutuamente. Enquanto um ciclo *while* é responsável pelo tratamento e representação sob forma de gráficos de dados resultantes de um determinado movimento, o segundo ciclo *while* é responsável pelo controlo do fluxo da *interface* com o utilizador. Na Figura 63 e 64 encontra-se uma imagem do primeiro e segundo ciclos *while*, respectivamente.

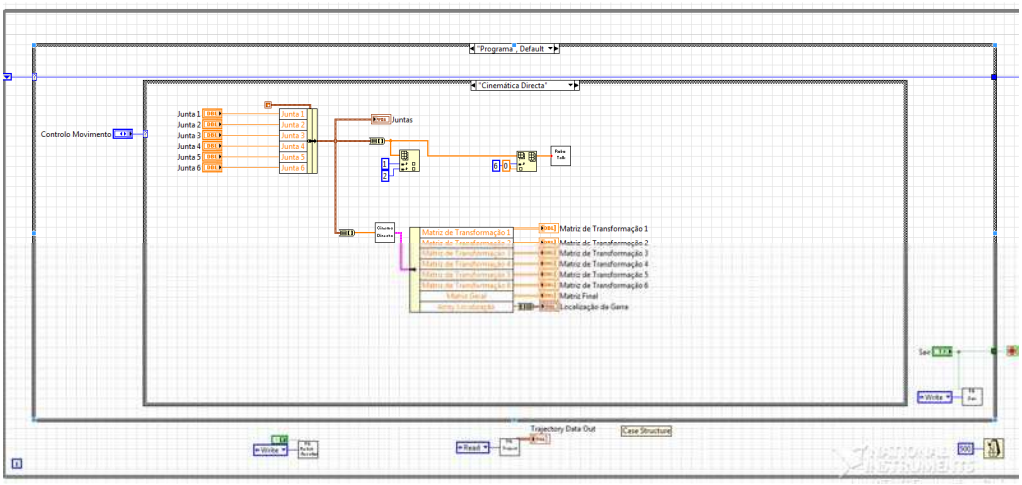

FIGURA 63 - DIAGRAMA DE BLOCOS DO CICLO *WHILE* RESPONSÁVEL PELO TRATAMENTO E REPRESENTAÇÃO DE DADOS

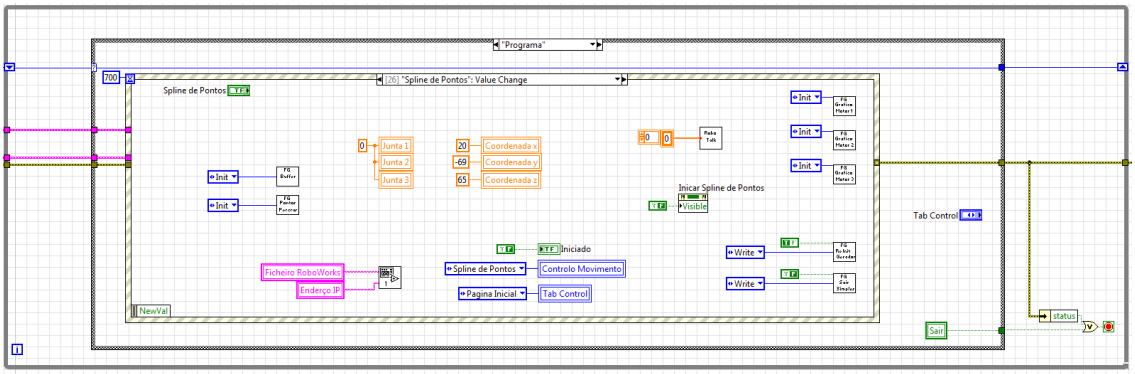

FIGURA 64 - DIAGRAMA DE BLOCOS DO CICLO *WHILE* RESPONSÁVEL PELO CONTROLO DE FLUXO DA INTERFACE

A execução do primeiro e segundo ciclos *while* está dividida em duas fases: Iniciação e Programa. Na primeira fase são inicializadas variáveis necessárias ao bom funcionamento destes ciclos. Na segunda fase de execução destes ciclos é executado o código propriamente dito.

A fase de programa do primeiro ciclo é constituída por um ciclo *case* constituído por sete casos: Parado, Cinemática Directa, Cinemática Inversa, Movimento Simples, Comando Manual, *Spline* de Pontos e Parâmetros de Controlo. Quando a aplicação é inicializada e este ciclo *while* entra na fase de programa, o código a executar é o que está contido no interior do caso Parado. Quando o utilizador prime um dos botões do menu raiz, o seu estado irá ser alterado mediante o botão que for premido. Todos os gráficos, controlos e indicadores numéricos presentes na interface são lidos ou escritos nesta fase do ciclo, onde são também usadas algumas *functions globals* para a aquisição e envio de informação para outros blocos da interface.

Quando o segundo ciclo *while* entra na fase de programa o código que é executado é basicamente uma estrutura de eventos. A estrutura de eventos definida contém no seu interior o código a executar quando o determinado botão do interface altera o seu valor ou estado. Desta maneira, quando o utilizador prime um qualquer botão na interface faz com que a variável associada ao botão altere o seu valor. Esta alteração irá fazer com que seja executado o código contido na estrutura de eventos referente ao evento sucedido.

Depois de explicado um pouco do funcionamento do primeiro e segundo ciclos *while* que controlam a interface com o utilizador irá agora ser apresentada a razão porque anteriormente foi dito que estes dois ciclos se complementam. Estes dois ciclos são como um condutor e um carro, se o carro precisa do condutor para andar, também o condutor precisa do carro para ele próprio se mover. Isto acontece uma vez que: o primeiro ciclo disponibiliza todo um fluxo de dados necessários à interface, enquanto o segundo ciclo controla as várias fases da interface, desta maneira o primeiro ciclo *while* será o carro enquanto o segundo ciclo será o condutor, sendo assim complementares.

5.5.4. Bloco gerador de trajectórias

Este bloco, tal como o nome indica, é responsável pelo planeamento de toda a trajectória do manipulador. Este bloco tem como parâmetros de entrada o *handle* dos três eixos que irão ser movimentados bem como o período de execução deste bloco. Um esquema do respectivo diagrama de blocos encontra-se representado na Figura 65.

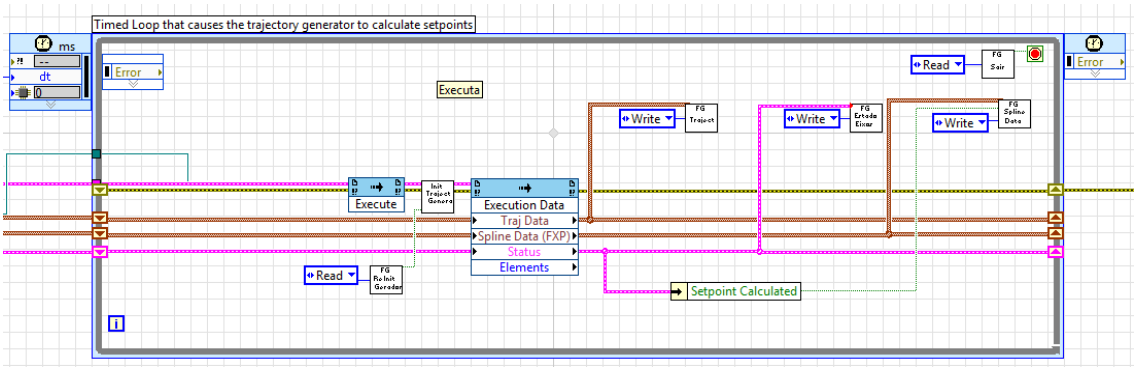

FIGURA 65 - DIAGRAMA DE BLOCOS DO GERADOR DE TRAJECTÓRIAS

Quando o utilizador inicia um movimento contínuo através da opção 'Movimento Simples' ou 'Trajecto', este bloco irá gerar *setpoints* com um período igual ao de execução do ciclo *while* de tempo real que constitui este bloco. Os *setpoints* gerados irão obedecer ao perfil de movimento que o utilizador definiu. Neste bloco encontram-se três *function globals* que são usadas para passar para os ciclos de interface e bloco da malha de controlo várias variáveis como por exemplo a posição e velocidade angular pretendida para o veio do motor, ou ainda o estado de execução do perfil de movimento.

À semelhança do que acontece nas *function globals* também neste bloco são usados *shift registers* como unidades de "memória". Neste bloco são usados três *shift registers* para guardar o valor de três *clusters* necessários às funções que calculam o novo *setpoint*. Embora não seja visível na Figura 65 os *shift registers* são também inicializados.

No que toca ao período de execução deste bloco, este terá que ser obrigatoriamente superior ao período de execução da malha de controlo para que o

manipulador tenha tempo de atingir o *setpoint* calculado da última vez que o bloco é executado. Desta maneira definiu-se um período de execução deste bloco de 80 ms.

Depois de o manipulador atingir a posição definida pelo utilizador, o bloco gerador de trajectórias irá continuar a ser executado. Contudo, as funções para gerar novos *setpoints* não irão gerar qualquer valor pois a posição atingida já foi alcançada.

5.5.5. Bloco da malha de controlo

O bloco da malha de controlo, tal como o nome sugere, é responsável pelo controlo de toda a trajectória do manipulador. Este bloco tem como parâmetros de entrada os *setpoints* calculados pelo bloco gerador de trajectórias ao longo do movimento do manipulador. Na Figura 66 encontra-se um extracto do respectivo diagrama de blocos.

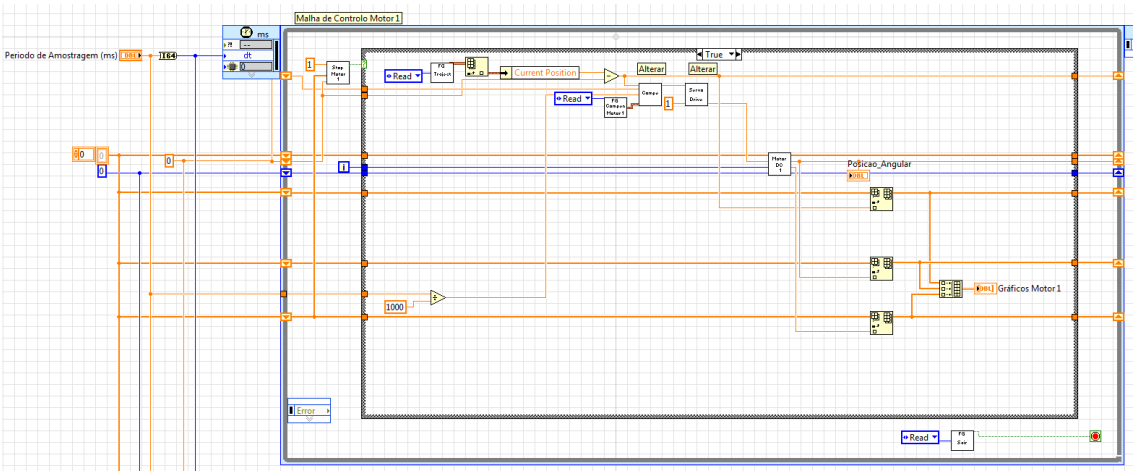

FIGURA 66 - DIAGRAMA DE BLOCOS DA MALHA DE CONTROLO

Embora na Figura 66 esteja apenas representado um único ciclo *while* de tempo real, o bloco da malha de controlo é na realidade constituído por mais dois ciclos idênticos ao apresentado na figura referida - um ciclo por cada junta do manipulador. Desta maneira apenas será explicado o funcionamento de apenas um destes três ciclos.

Foi referido que este o bloco da malha de controlo teria que ser executado a uma cadência maior que a do bloco gerador de trajectórias para que o manipulador tivesse

tempo para atingir o *setpoint* calculado durante a execução deste último bloco. Desta maneira foi definido um período de execução de 20 ms para cada um dos três ciclos do que constituem o bloco da malha de controlo.

No interior do ciclo *while* de tempo real encontra-se um ciclo *case* que é controlado por um outro bloco de nome "Stop_Motor_1.vi". O nome desta função varia consoante a junta do manipulador, por exemplo, no caso da junta dois, o ciclo case será controlado pelo bloco "Stop_Motor_2.vi". Este bloco tem uma variável de saída do tipo booleana. Desta maneira quando a variável de saída deste bloco é *true* o manipulador está em movimento. Pelo contrário quando a variável de saída assume o valor de *false* o manipulador encontra-se parado tendo atingido o ponto pretendido. Assim, quando o manipulador se encontra parado, embora o ciclo *while* seja executado com a periodicidade de 20 ms, o código referente à malha de controlo propriamente dita não é executado. Esta estratégia revelou-se fundamental para não sobrecarregar o processador do computador onde a aplicação está a ser executada. A criação de um bloco de controlo para o ciclo *case* de cada motor permite poupar alguma capacidade de processamento do processador, pois desta maneira apenas é executada a malha de controlo dos eixos que se encontrem em efectivo movimento.

Depois do exposto, conclui-se que o código referente à malha de controlo se encontra, assim, no interior do ciclo *case* apresentado no último parágrafo. A malha de controlo implementada é baseada do controlo da posição angular do veio do motor DC da junta referente ao ciclo *while* onde esta está a ser executada a malha. Na Figura 67 encontra-se o diagrama de blocos da malha de controlo implementada em todas as juntas do manipulador.

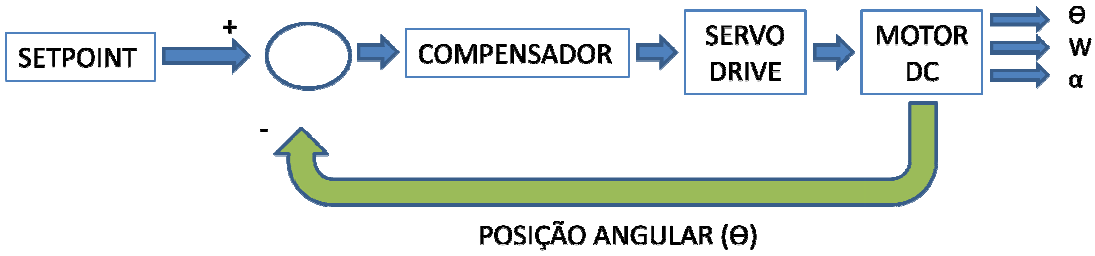

FIGURA 67 - MALHA DE CONTROLO IMPLEMENTADA NO SIMULADOR

Tal como se pode observar na Figura 67 esta malha de controlo é constituída por três blocos: Compensador, Servo Drive e Motor DC. Os dois últimos blocos pretendem simular o equipamento que nunca se chegou a utilizar no manipulador real que fora disponibilizado. O funcionamento destes blocos será explicado em pontos seguintes desta dissertação.

À semelhança de outros blocos também neste são usados *shift registers* como unidade de memória. Para a implementação desta malha de controlo no total foram utilizados sete *shift registers* de vários tipos: um para guardar um *array* que contém todas as posições angulares do veio do motor, um *array* que regista o valor da velocidade angular do veio do motor, um *array* que regista a aceleração angular do veio do motor, um *array* que regista todas as entradas que são enviadas para o bloco que simula o comportamento do motor, um guarda o sinal de erro (diferença entre o *setpoint* e posição angular do veio do motor) da última execução da malha, um guarda a posição angular do veio do motor da última execução da malha de controlo e, por último, um que guarda o número de iterações do ciclo *while* desde a última vez que a malha de controlo foi executada. Os três primeiros *array's* são usados para a construção de um gráfico onde seja possível uma visualização mais detalhada da evolução da posição, velocidade e aceleração angular. Os restantes *shift registers* são utilizados para o cálculo e validação da malha de controlo.

5.5.5.1. Bloco do compensador

Este bloco tem como entrada o sinal de erro (resultado da diferença entre o *setpoint* e o sinal de feedback proveniente do bloco de simulação do motor) e tem como saída um sinal em tensão que irá para o bloco de simulação do servo drive. Tal como se observa na Figura 67 este bloco tem ainda à entrada um cluster proveniente de uma *function global*, 'FG_Compensador_Motor_1.vi'. Esse cluster é constituído por uma variável que define o tipo de compensador que o utilizador escolheu através da interface, bem como as constantes desse compensador (Ki, Kp e Kd). Na Figura 68 pode-se observar mais em detalhe o diagrama de blocos implementado.

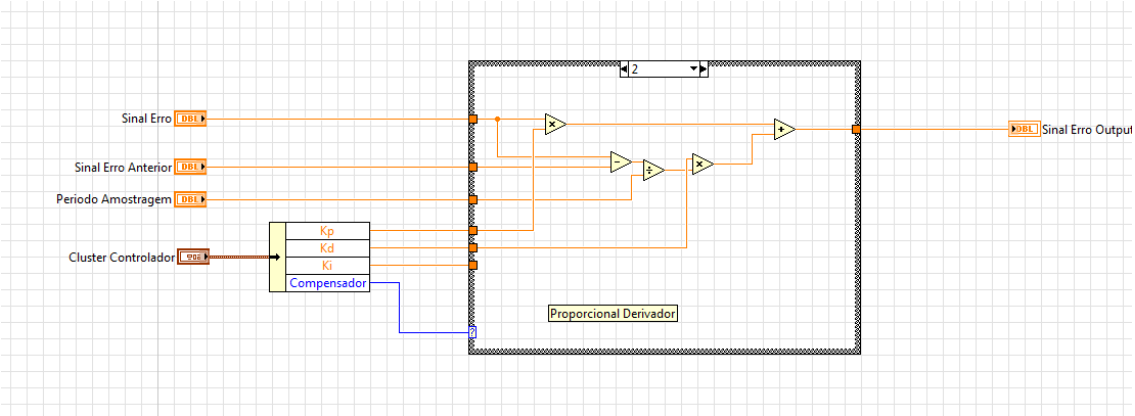

FIGURA 68 - DIAGRAMA DE BLOCOS DO COMPENSADOR

Tal como se pode ver na figura anterior, este bloco é essencialmente formado por um ciclo *case*. Dependendo do compensador escolhido pelo utilizador assim também depende o caso deste ciclo que é executado. Foram definidos quatro tipos de compensadores: proporcional (P), proporcional derivador (PD), proporcional integrador (PD), e proporcional integrador derivador (PID). De seguida vão ser apresentadas as equações utilizadas para implementar os vários tipos de compensador.

$$P : Output_n = Input_n * Kp \qquad \text{Equação 50}$$

$$PD : Output_n = Input_n * Kp + \frac{Input_n - Input_{n-1}}{dt} Kd$$

$$PI : Output_n = Input_n * Kp + \frac{Input_n - Input_{n-1}}{2.Ki}$$

$$PID : Output_n = Input_n * Kp + \frac{Input_n - Input_{n-1}}{2.Ki} + \frac{Input_n - Input_{n-1}}{dt} Kd$$

Nas equações anteriores *input* define o sinal de erro que entra neste bloco, *output* define o sinal em tensão que irá ser enviado para o bloco de simulação do servo drive, *n* define o número da amostra actual e *dt* o período de amostragem utilizado, neste caso este será igual ao período de execução da malha de controlo, ou seja 20 ms.

5.5.5.2. *Bloco do servo drive*

Este bloco tem como parâmetro de entrada um sinal em tensão proveniente do compensador e como parâmetro de saída um sinal também ele em tensão para o bloco de simulação do motor. Este bloco é também vulgarmente conhecido como amplificador uma vez que tem por missão amplificar um sinal em tensão. Desta maneira o que este bloco faz é justamente isso, coloca na saída o produto do sinal da entrada com uma constante a definir.

5.5.5.3. *Bloco de simulação do motor*

Este bloco tem como parâmetros de entrada: um sinal em tensão proveniente do bloco do servo drive, um variável que indica o tempo já decorrido deste o início do movimento e ainda um *array* que tem registado todos os valores provenientes do bloco do servo drive, *array* esse denominado de *array* das entradas. No que se refere às saídas, este bloco apresenta cinco: posição angular actual, velocidade angular actual, aceleração angular actual, número de iterações do ciclo decorridas desde o início do movimento e ainda o *array* das entradas já acrescido com o último valor enviado pelo bloco do servo drive. Neste bloco é simulado o comportamento de um motor DC sem escovas. O diagrama de blocos respectivo é apresentado na Figura 69.

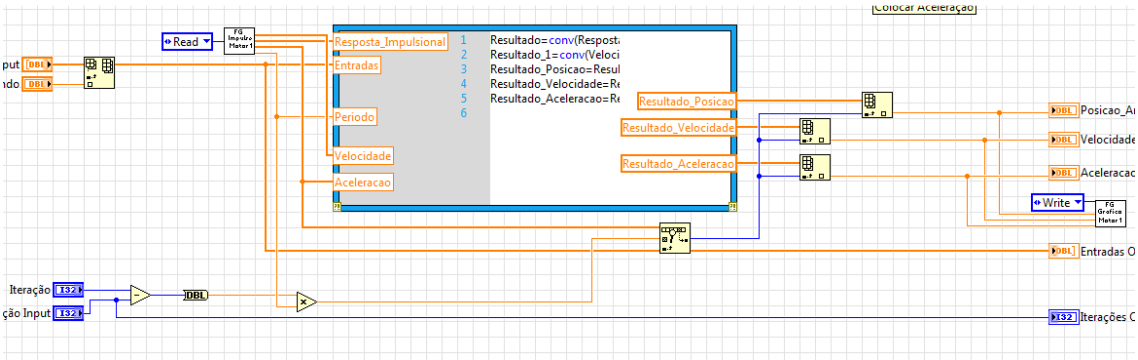

FIGURA 69 - DIAGRAMA DE BLOCOS DE SIMULAÇÃO DO MOTOR

O bloco de simulação do motor possui como parâmetros de entrada: uma variável denominada de sinal de comando, que contém o valor enviado pelo bloco do servo drive, um *array* que tem registado todos os valores desta última variável até esse momento, uma variável que contém o número de iterações do ciclo *while* deste que a

aplicação foi iniciada e ainda uma variável que contém o numero de iterações do mesmo ciclo *while* até momento em que o manipulador entrou em movimento e consequente execução da malha de controlo. Como parâmetros de saída apresenta: um *array* que contém todos os registos da variável de comando desde o início do ciclo, uma variável que contém o valor da posição angular do motor, uma variável que contém o valor da velocidade angular, uma variável que contém o valor da aceleração angular e ainda uma variável que contém o número de iterações do ciclo *while* ocorridas até ao momento em que o manipulador iniciou o um movimento segundo o eixo correspondente da malha de controlo. São lidos também quatro *array's*: três que contêm a resposta impulsional das funções de transferência que relacionam a posição, velocidade e aceleração angular com a tensão colocada aos terminais motor que é controlado naquela malha de controlo, e um que contém todos os instantes em que as amostras dos três primeiros *array's* foram efectuadas. Estes quatro *array's* são provenientes da *function global* 'FG_Resposta_Impulsinal_Motor_1.vi' que é inicializada com estes valores no bloco de inicialização da aplicação. Através da mesma é também passado o valor usado como período de amostragem na inicialização dos três *array's* anteriormente referidos. Na Figura 70 encontra-se uma ilustração de como é processado o cálculo da posição, velocidade e aceleração angular.

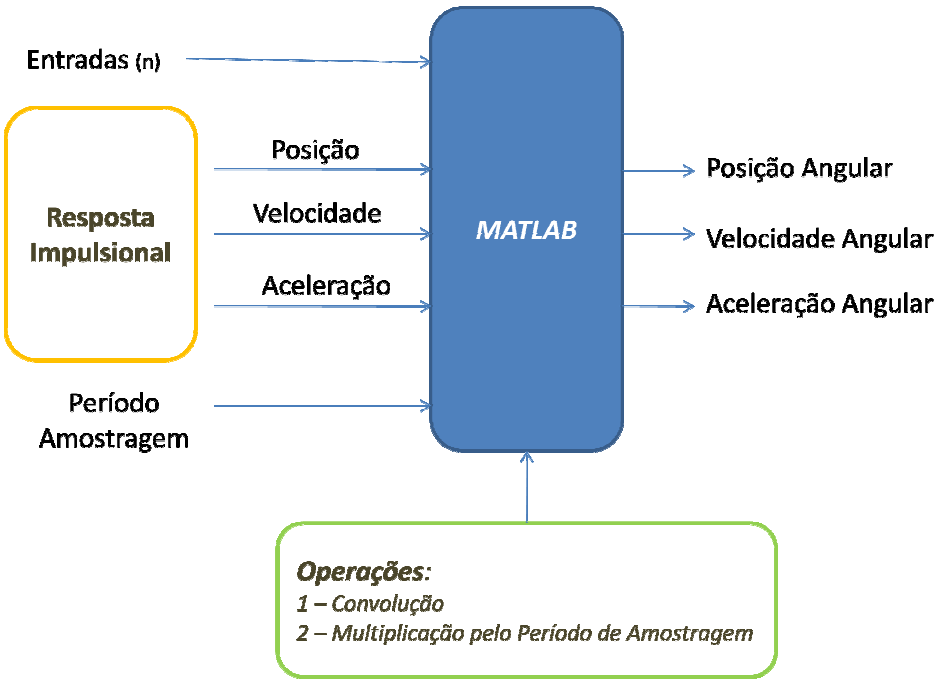

FIGURA 70 - DIAGRAMA DE FUNCIONAMENTO DO BLOCO DE SIMULAÇÃO DO MOTOR DC

O núcleo deste bloco de simulação do motor reside no trecho que código que está contido no interior do ciclo a executar no *Matlab*. Na Figura 69 é representado pelo ciclo delimitado por uma linha de cor azul. Esse código recebe como parâmetros de entrada um *array* que contém o valor actual do sinal de comando bem como todos os valores passados deste, três dos *array's* passados pela *function global* 'FG_Resposta_Impulsinal_Motor_1.vi' que contém respostas impulsionais das funções de transferência (calculadas no bloco 'Iniciacao.vi') e ainda o valor do período de amostragem passada pela mesma *function global*. No interior deste ciclo são calculados três novos *array's*. Estes novos array's resultam da convolução do *array* que regista todos os valores da variável de comando e os três *array's* passados pela já referida *function global*. No fim de efectuadas as três convoluções é efectuada a multiplicação dos três *array's* resultantes pelo período de amostragem já mencionado, *l*, desta maneira todas as operações, inclusive a convolução, são efectuadas no *Matlab*. No fim desta multiplicação o *Matlab* devolve ao *Labview* três *array's* que contêm: o registo de todo o histórico de posições angulares do veio do motor controlado, o registo de todo o histórico de velocidades angulares do veio do motor controlado e o registo de todo o histórico de acelerações angulares do motor controlado.

Uma vez que se pretende que este bloco apresente como parâmetros de saída o valor da posição, velocidade e aceleração angulares, é necessário conhecer que índice dos array's enviados pelo *Matlab* corresponde ao momento da simulação.

A variável que contém o número de iteração desde o início da aplicação e a que contém o número de iterações são usadas para calcular o instante em que a simulação ocorre. Este é calculado através da subtracção da segunda à primeira variável. O resultado da subtracção é de seguida multiplicado pelo período de amostragem proveniente da já referida *funtion global*. O resultado deste produto irá ser utilizado para pesquisar no *array* que contém todos os instantes de tempo em que foram feitas as amostragens das respostas impulsionais. O índice do *array* que contiver um instante igual ao instante actual da simulação irá corresponde ao índice dos três array que contém o histórico da posição, velocidade e aceleração angular, do valor actual destas

três variáveis. É assim calculado valor da posição, velocidade e aceleração angular actual do veio do motor.

5.5.5.4. Funções de transferência do motor

Depois de no capítulo anterior se ter feito uma pequena apresentação deste tipo de motores, neste ponto da dissertação será deduzido um modelo matemático capaz de traduzir o comportamento deste considerando que é controlado pela corrente do induzido. De seguida será então calculada a função de transferência deste tipo de motores.

O motor DC é um dispositivo que produz um movimento de rotação do seu eixo quando lhe é aplicada uma tensão contínua, V_m , aos seus terminais. Essa tensão de alimentação, V_m, irá criar uma corrente i_a em cada bobine que constitui o estator do motor. A passagem da corrente i_a irá originar uma força que será responsável por fazer rodar o veio do motor. As forças produzidas têm o nome de forças de *Lorentz*, e são proporcionais a i_a*B, onde B representa o campo magnético produzido pelos ímanes permanentes contidos no veio do motor. Na Figura 71 encontra-se o circuito equivalente eléctrico de um motor deste tipo.

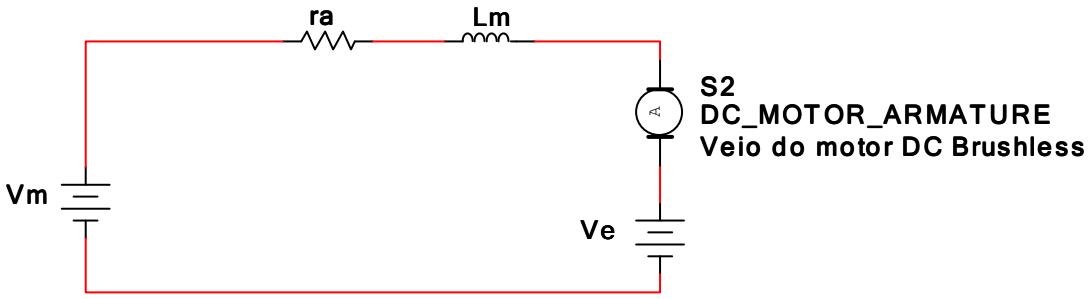

FIGURA 71 - CIRCUITO EQUIVALENTE ELÉCTRICIO DE UM MOTOR DC CONTROLADO PELO INDUZIDO

O movimento de rotação do motor por sua vez irá conduzir originar uma tensão, V_e, contrária à tensão de alimentação do motor. Essa tensão tem o nome de força contra-electromotriz e é igual à Equação 51 onde *w* representa a velocidade angular de rotação do veio do motor e k_b é uma constante que depende do número de espiras do motor e da intensidade do campo magnético.

$$V_e = k_b.w$$ Equação 51

Depois de explicado um pouco do funcionamento do motor vão agora ser apresentadas as deduções matemáticas que permitem calcular as três funções de transferência já mencionadas.

Função de transferência que relaciona a velocidade angular com a tensão de alimentação ($\frac{w(s)}{V_m(s)}$):

Uma vez que o binário desenvolvido pelo motor é proporcional à corrente que passa pelas suas espiras então este é igual à Equação 52, onde K_m é uma constante que depende de parâmetros físicos do motor.

$$T_m = K_m.i_a$$ Equação 52

O binário de um motor pode também ser calculado através da Equação 53, onde *J* representa o momento de inércia do veio do motor e *f* o atrito do mesmo. Se se passar as equações 52 e 53 para o domínio de *Laplace* e de seguida se igualarem pode-se calcular a corrente i_a através da Equação 54.

$$T_m = J\dot{w} + fw$$ Equação 53

$$I_a(s) = \frac{W(s)}{k_m}(sJ + f)$$ Equação 54

Aplicando as leis de *Kirchoff* ao circuito eléctrico equivalente do motor, presente na Figura 71, pode se chegar á Equação 55.

$$V_m = r_a i_a + L_a \frac{\partial i_a}{\partial t} + V_e$$ Equação 55

Substituindo V_e pela Equação 51 e passando a Equação 55 para o domino de *Laplace* obtém-se a Equação 56.

$$V_m = (r_a + sL_a)I_a(s) + k_b W(s)$$ Equação 56

Substituindo a Equação 54 na Equação 56 obtém então a função de transferência que relaciona a velocidade angular, *w* com a tensão de alimentação do motor V_m. Essa mesma função de transferência está representada na Equação 57.

$$\frac{W(s)}{I_a(s)} = \frac{k_m}{s^2 L_a J + s(L_a f + r_a J) + k_m k_b + r_a f}$$ Equação 57

Função de transferência que relaciona a aceleração angular com a tensão de alimentação ($\frac{\alpha(s)}{V_m(s)}$):

Sabendo da física que a aceleração de um qualquer corpo pode ser calculada através da derivada da sua velocidade, pode-se dizer que a Equação 58 enuncia uma proposição verdadeira considerando as condições iniciais nulas.

$$\frac{\alpha(s)}{V_m(s)} = s\frac{W(s)}{V_m(s)}$$ Equação 58

Desta maneira a função de transferência que relaciona a aceleração angular com a tensão de alimentação do motor é igual à Equação 59.

$$\frac{\alpha(s)}{V_m(s)} = \frac{sk_m}{s^2 L_a J + s(L_a f + r_a J) + k_m k_b + r_a f}$$ Equação 59

Função de transferência que relaciona a aceleração angular com a tensão de alimentação ($\frac{\theta(s)}{V_m(s)}$):

Recorrendo novamente à física, sabe-se que a posição de um determinado corpo é igual ao integral da sua velocidade, desta maneira pode-se dizer que a Equação 60 enuncia uma proposição verdadeira considerando novamente condições iniciais nulas.

$$\frac{\theta(s)}{w(s)} = \frac{W(s)}{sV_m(s)}$$ Equação 60

Através da igualdade representada na Equação 60 pode-se então calcular a função de transferência que relaciona a posição angular do veio do motor com a tensão de alimentação do mesmo, V_m. Essa mesma função de transferência está presente na Equação 61.

$$\frac{\theta(s)}{V_m(s)} = \frac{k_m}{s^3 L_a J + s^2 (L_a f + r_a J) + s(k_m k_b + r_a f)}$$ Equação 61

Para todos os motores foram usadas as seguintes constantes:

$r_a = 1{,}3\Omega$
$L_a = 0{,}2H$
$k_b = 0{,}1$
$k_m = 1$
$f = 0{,}1N$

Os momentos de inércia dos três motores sem carga foram os seguintes:

$Motor1 : J_0 = 1{,}58(k_g cm^2)$
$Motor2 : J_0 = 3{,}25(k_g cm^2)$
$Motor3 : J_0 = 0{,}7(k_g cm^2)$

No caso de um manipulador real a estes momentos de inércia seria acrescido o momento de inércia da carga a que cada motor está acoplado. Essa mesma carga seria dinâmica consoante o movimento do robot uma vez que o ângulo da carga com o chão irá variar o que origina uma variação da força peso. Desta maneira o momento de inércia da carga varia ao longo do movimento do robot.

5.5.5.5. Bloco de comunicações com o RoboWorks

A comunicação entre a aplicação e o *RoboWorks* foi efectua usando para tal uma biblioteca de funções disponibilizada já pelo fabricante deste software. A referida

biblioteca tem o nome de 'robotalk.llb'. Na Figura 72 é possível visualizar todas as funções disponibilizadas nesta biblioteca.

FIGURA 72 – BIBLIOTECA DE FUNÇÕES USADAS PARA COMUNICAR COM O *ROBOWORKS*

Tal como se pode ver na figura anterior, a biblioteca utilizada é constituída por quatro funções: 'Connect.vi', 'Disconnect.vi', 'GetTagValues.vi' e SetTagValues.vi'. Enquanto a primeira função é usada para estabelecer a ligação entre a aplicação e o *RoboWorks*, a segunda é utilizada para terminar essa mesma ligação. A terceira função só pode ser usada depois de se já ter estabelecida uma ligação com o *RoboWorks* e é usada para saber qual é a posição actual das várias juntas do manipulador. A quarta e última função também só pode ser usada depois de já se ter estabelecida uma ligação com o *RoboWorks*, e é usada para transmitir ao manipulador a posição desejada para as suas juntas.

Para se estabelecer uma ligação com o software de simulação é apenas necessário introduzir o nome do simulador que se pretende simular, no caso 'PUMA760Robot' e o endereço *IP* da máquina onde este está a ser executado. Desta maneira é possível executar a aplicação projectada num computador que esteja ligado através de TCP/IP, a um outro computador onde esteja a ser executado o *RoboWorks*.

Na aplicação, a ligação ao *RoboWorks* é feita quando o utilizador clica em qualquer um dos botões presentes no menu raiz, à excepção do botão de 'Parâmetros de

Controlo'. A mesma ligação é terminada quando o utilizador prime o botão de 'Página Inicial' e retorna ao menu raiz da aplicação.

Tal como já foi dito anteriormente, este software de visualização não apresenta qualquer tipo de dinâmica associada ao manipulador. Desta maneira apenas foi usada a função 'SetTagValues.vi' na aplicação uma vez que não se justificava pelo motivo já avançado o uso da função da aquisição do valor actual das juntas do manipulador. Na aplicação, a função de envio de dados para o manipulador é usada de duas maneiras distintas. Quando o utilizador prime o botão de 'Cinemática Directa', 'Comando Manual' ou 'Cinemática Inversa' no menu raiz da aplicação, esta função é chamada cada vez que o utilizador introduz uma nova coordenada na interface. Quando o utilizador no menu raiz opta pela opção de 'Movimento Simples' ou 'Trajecto' são efectuados movimentos contínuos onde o manipulador adquire dinâmica. Desta maneira são enviados dados para o *RoboWorks* cada vez que o bloco da malha de controlo é executada.

5.5.5.6. Bloco da cinemática directa do manipulador PUMA 760

À semelhança do manipulador que se pretendia controlar na dissertação também para este foi necessário calcular a cinemática inversa de cinemática directa. Anteriormente já foi referido que o primeiro passo a dar para calcular a cinemática do manipulador é definir a tabela de parâmetros da cinemática do manipulador. A Tabela 3 apresenta os parâmetros da cinemática do manipulador do simulador, no caso um PUMA 760. Esta tabela não segue a convenção utilizada para calcular a cinemática do manipulador que tinha à disposição uma vez que foi disponibilizada pelo modelo do *RoboWorks*. A convenção considerada segue a utilizada por John Craig. [9]

TABELA 3 - PARÂMETROS DA CINEMÁTICA DO MANIPULADOR PUMA 760

Elo	$\alpha_{(i-1)}$	$a_{(i-1)}$	d_i	Θ_i
Elo_{0-1}	0	0	0	Θ_1
Elo_{1-2}	-90	0	0	Θ_2
Elo_{2-3}	0	65	20	Θ_3
Elo_{3-4}	-90	0	69	Θ_4
Elo_{4-5}	90	0	0	Θ_5
Elo_{5-6}	-90	0	0	Θ_6

A matriz de transformação utilizada segundo esta convenção é a descrita na Equação 62.

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos(\theta_i) & -sen(\theta_i) & 0 & a_{i-1} \\ sen(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -sen(\alpha_{i-1}) & -d_i sen(\alpha_{i-1}) \\ sen(\theta_i)sen(\alpha_{i-1}) & \cos(\theta_i)sen(\alpha_{i-1}) & \cos(\alpha_{i-1}) & d_i \cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} $$ Equação 62

A matriz da Equação 62 foi utilizada no bloco 'Cinemática_Directa.vi' para calcular as sucessivas matrizes de transformação ao longo do manipulador. Na Figura 73 encontra-se uma imagem do diagrama de blocos deste mesmo bloco.

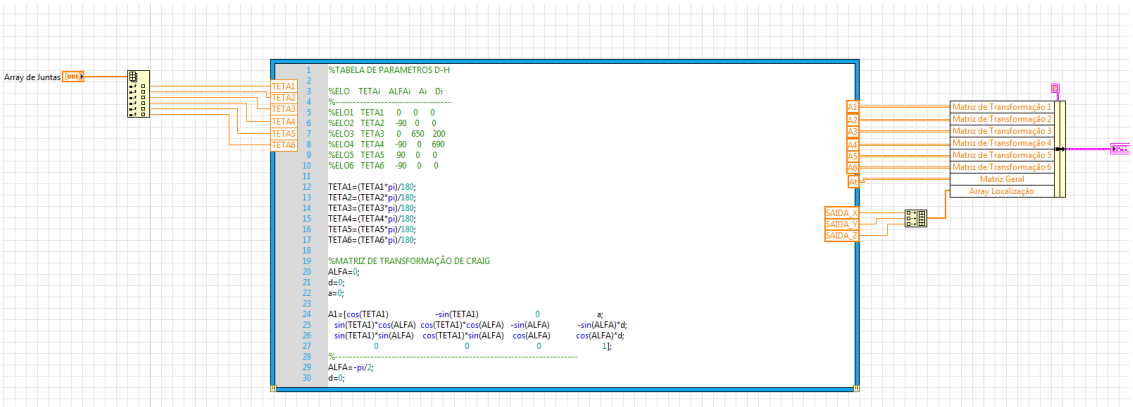

FIGURA 73 - DIAGRAMA DE BLOCOS DO BLOCO 'CINEMATICA_DIRECTA.VI'

Tal como no bloco de simulação do motor o núcleo deste bloco reside no bloco delimitado a azul na Figura 73 que executa o código contido no seu interior no servidor *Matlab*. O código que é executado no servidor *Matlab* tem como parâmetros de entrada o valor da posição angular das seis juntas do manipulador. Como parâmetros de saída apresenta as várias matrizes de transformação dos vários elos do manipulador, bem como a matriz de transformação que relaciona o referencial da base do manipulador com o referencial da garra do manipulador. O servidor *Matlab* ainda envia as coordenadas cartesianas para localização espacial da garra do manipulador.

O código executado no servidor *Matlab* utiliza os parâmetros da cinemática definidos na Tabela 3 e a Equação 62 para calcular a matriz de transformação referente a cada elo do manipulador. É calculada ainda a matriz de transformação que relaciona o referencial da base com o referencial da garra do manipulador através do produto das sucessivas matrizes de transformação do manipulador.

Este bloco que calcula a cinemática directa do manipulador PUMA 760 é utilizado um pouco por toda a aplicação para efectuar cálculos da localização espacial da garra. As sucessivas matrizes de transformação, bem como a matrizes de transformação geral, podem ser observadas se o utilizador no menu raiz da aplicação clicar no botão 'Cinemática Directa' e de seguida premir o botão 'Matrizes de Transformação'.

5.5.5.7. Bloco da cinemática inversa do manipulador PUMA 760

À semelhança da cinemática directa, também os cálculos da cinemática inversa do manipulador foram baseados no livro de John Craig. [9] Com base neste livro foram obtidas as funções que foram implementadas no bloco que calcula a cinemática inversa do manipulador. As expressões obtidas tiveram que ser adaptadas ao sistema de eixos da base que está implementado no *RoboWorks*. A troca de eixos efectuada foi a seguinte:

$$p_x \rightarrow RoboWorks : p_z$$
$$p_y \rightarrow RoboWorks : p_x$$
$$p_z \rightarrow RoboWorks : p_y$$

As funções usadas para calcular a cinemática inversa foram então as seguintes:

A expressão que permite calcular a posição angular do veio do primeiro motor:

$$\theta_1 = a\tan 2(p_x, p_y) - a\tan 2(d_3 \pm \sqrt{p_z^2 + p_x^2 - d_3^2}$$ Equação 63

A expressão que permite calcular a posição angular do veio do terceiro motor:

$$K = \frac{p_z^2 + p_x^2 + p_y^2 - a_2^2 - a_3^2 - d_3^2 - d_4^2}{2a_2}$$

$$\theta_3 = a\tan 2(a_3, d_4) - a\tan 2(K, \pm\sqrt{a_3^2 + d_4^2 - K^2})$$ Equação 64

A expressão que permite calcular a posição angular do veio do segundo motor:

$$\cos(\theta_{23}) = \frac{(a_2 sen(\theta_3) - d_4)p_y + (a_3 + a_2\cos(\theta_3))(\cos(\theta_1)p_z + sen(\theta_1)p_x)}{p_y^2 + (\cos(\theta_1)p_z + sen(\theta_1)p_x)^2}$$

$$sen(\theta_{23}) = \frac{(-a_3 - a_2\cos(\theta_3))p_y + (\cos(\theta_1)p_z + sen(\theta_1)p_x)(a_2 sen(\theta_3) - d_4)}{p_y^2 + (\cos(\theta_1)p_z + sen(\theta_1)p_x)^2}$$

$$\theta_{23} = a\tan 2(sen(\theta_{23}), \cos(\theta_{23}))$$

$$\theta_2 = \theta_{23} - \theta_3$$ Equação 65

Depois de apresentadas as equações que permitem calcular a cinemática inversa do manipulador vai-se passar à apresentação do diagrama de bloco do bloco que implementa tal cinemática. O bloco que tem tal missão é o bloco 'Cinematica_Inversa.vi' sendo o seu diagrama de blocos apresentado na Figura 74.

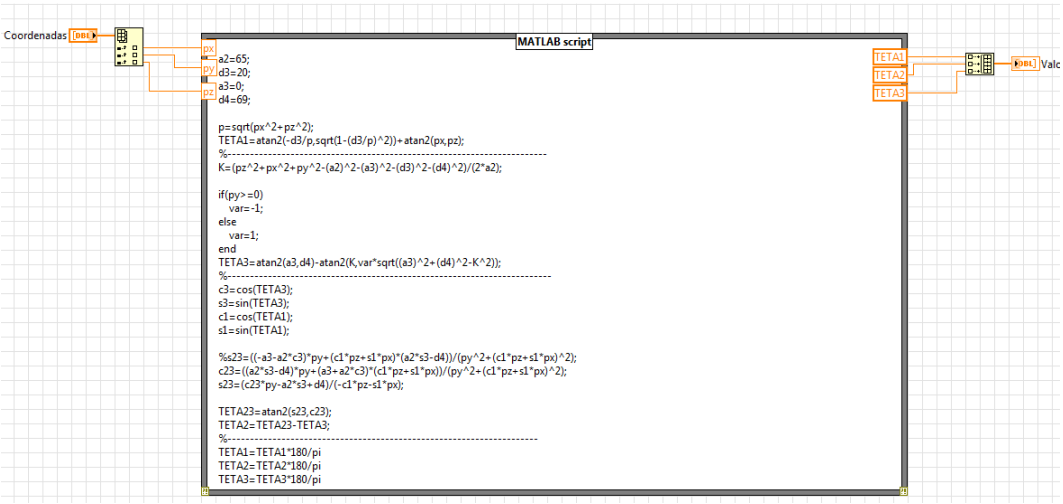

FIGURA 74 - DIAGRAMA DE BLOCOS DO BLOCO 'CINEMATICA_INVERSA.VI'

À semelhança do bloco da cinemática directa apresentado no ponto anterior desta dissertação o núcleo deste bloco reside no bloco que é executado no servidor de *Matlab*. Esse código encontra-se no interior do bloco delimitado a cinzento na Figura 74.

Este bloco tem como parâmetros de entrada o valor das três coordenadas cartesianas e tem como parâmetros de saída o valor da posição angular correspondente das três primeiras juntas.

Este bloco e o bloco da cinemática directa são blocos vitais para o correcto funcionamento do manipulador sendo executados bastantes vezes ao longo da aplicação. Por último falta apenas referir que quando o utilizador prime no menu raiz o botão 'Cinemática Inversa' ou 'Cinemática Directa' e indica um ponto pretendido estes dois blocos comunicam directamente com o bloco de comunicações do *RoboWorks*.

O principal objectivo desta dissertação visava o controlo de movimentos de um manipulador de seis eixos. Para se efectuar o controlo de movimentos do referido manipulador era necessário programar toda uma aplicação capaz de efectuar o referido controlo, bem como uma interface simples e amiga do utilizador. Toda a programação da aplicação devia ser feita em *Labview*.

O manipulador de seis eixos que se pretendia controlar era proprietário da empresa Selmatron, sendo que este já possuía motores que permitissem o seu movimento. Logo, na fase inicial foram encontrados problemas de funcionamento nos servo drives que controlavam os motores do robot. Para solucionar esta dificuldade a empresa detentora do manipulador achou por bem preceder à aquisição de novo equipamento.

Em fases diferentes foram equacionadas duas soluções possíveis: a aquisição de servo drives AX 5000 e motores AM3000 da empresa *Beckhoff* e ainda a aquisição de servo drives da serie MSD da empresa *MOOG*. A primeira solução tentada revelou-se infrutífera uma vez que a linguagem de programação que se pretendia usar, *Labview*, ainda não suportava o protocolo de comunicações *SERCOS*. A segunda solução equacionada não foi testada pois o equipamento que eu seleccionei não chegou à

empresa no devido tempo, não chegando em tempo útil para esta dissertação. Estas sucessivas alterações e a constante falta de material vital para o desenvolvimento da dissertação colocaram grandes dificuldades na progressão programada para o trabalho pois, por um lado, exigiram o estudo aprofundado do hardware e dos protocolos de comunicação específicos (assim como o desenvolvimento de código para os implementar e testar) e, por outro lado, não permitiram que se viesse a testar com equipamento real todo o software de interface e de controlo desenvolvido.

Em determinado ponto da dissertação comecei a procurar alternativas à no campo da simulação. Após grande procura encontrei o software de simulação *RoboWorks*. Este software permitiu assim a construção de uma aplicação que tentei que fosse o mais próxima possível da realidade.

5.6.Resultados

Para testar a aplicação efectuou-se o teste de mover o manipulador até um determinado local partindo este da posição inicial. Assim no menu raiz da aplicação clicou-se no botão 'Movimento Simples' e introduziram-se as seguintes coordenadas: Teta 1 = 180^{o}, Teta 2 = -45^{0} e ainda Teta 3 = -45^{o}. Durante o movimento recolheram-se dados provenientes da malha de controlo, presentes nas figuras a baixo apresentadas. Falta apenas referir que a malha de controlo já possuía um compensador derivativo com os parâmetros (K_p=0,1 e K_d=10).

Dados obtidos durante o movimento do manipulador:

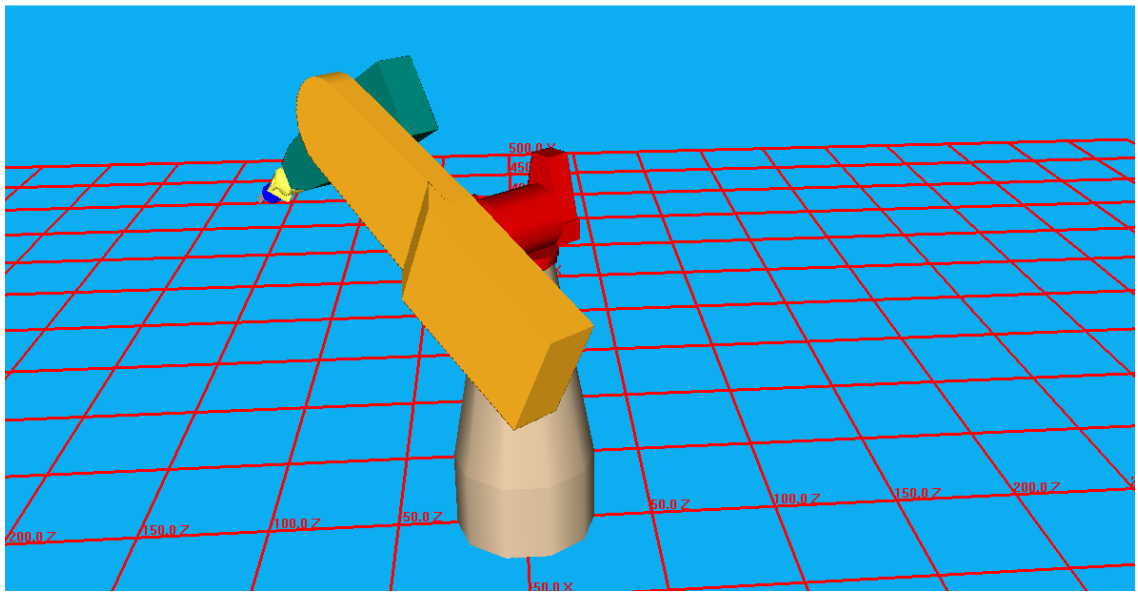

FIGURA 75 - MANIPULADOR PUMA 760 DURANTE O MOVIMENTO

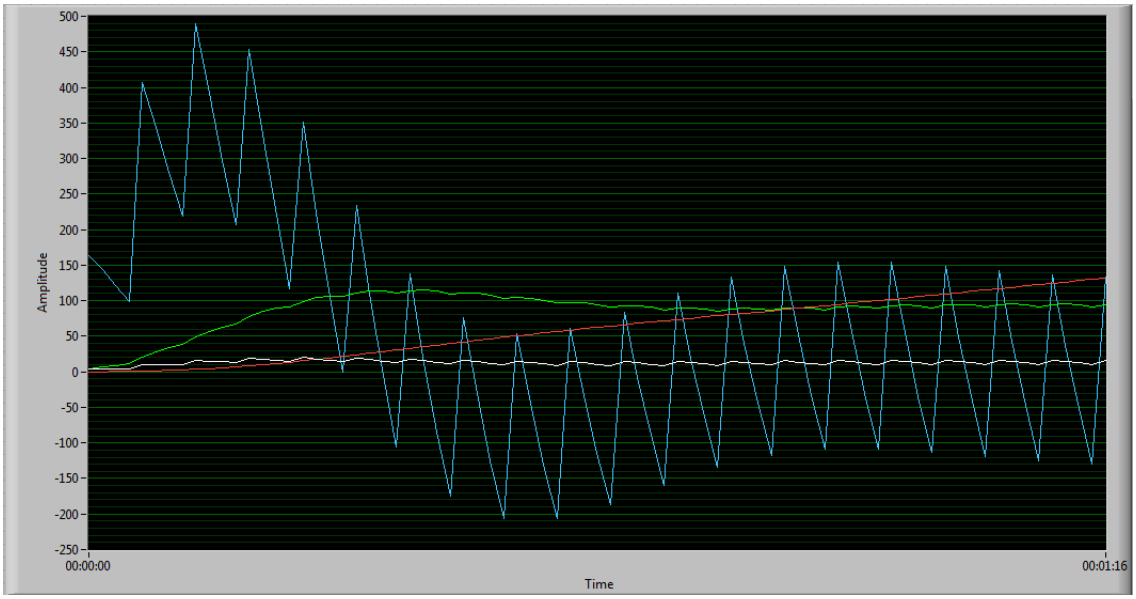

FIGURA 76 - GRÁFICO DE DADOS SOBRE MOTOR 1 (TETA 1 = 131^{o})

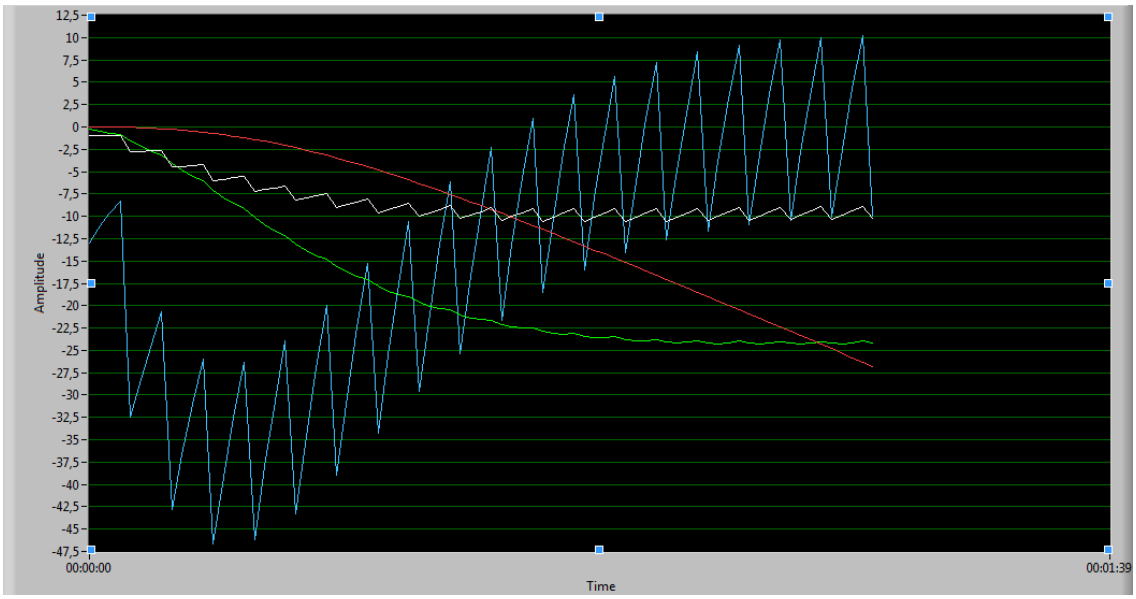

FIGURA 77 - GRÁFICO DE DADOS SOBRE MOTOR 2 (TETA 2 = - 27^{O})

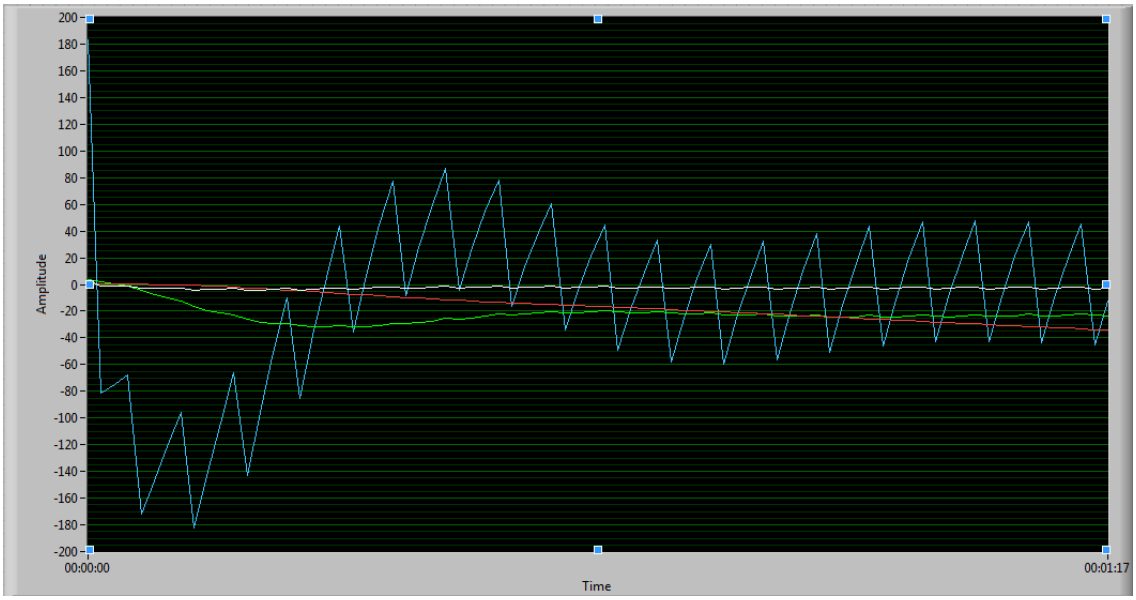

FIGURA 78 - GRÁFICO DE DADOS SOBRE MOTOR 3 (TETA 3 = - 34^{O})

Dados recolhidos quando o manipulador atinge a posição pretendida:

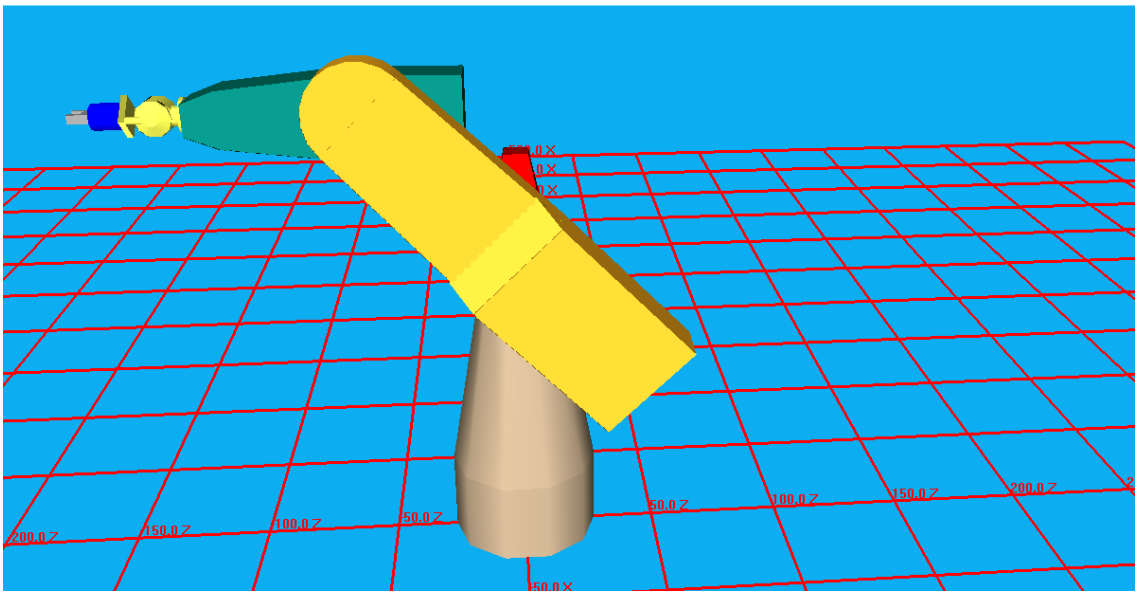

FIGURA 79 - MANIPULADOR PUMA 760 NA POSIÇÃO FINAL PRETENDIDA

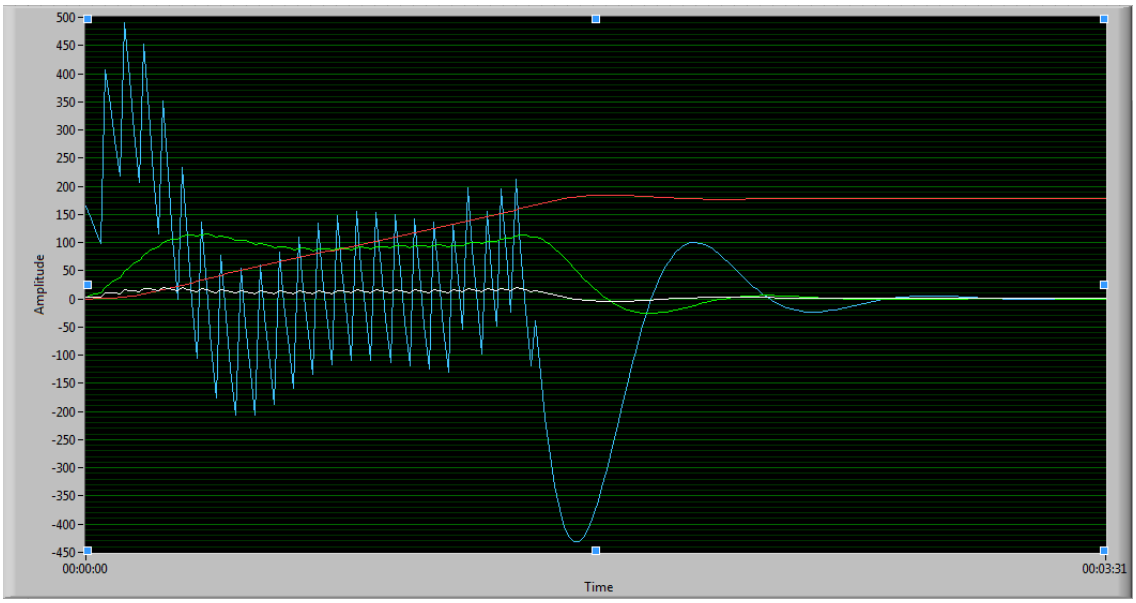

FIGURA 80 - GRÁFICO DE DADOS SOBRE MOTOR 1 (TETA 1 = 180^{o})

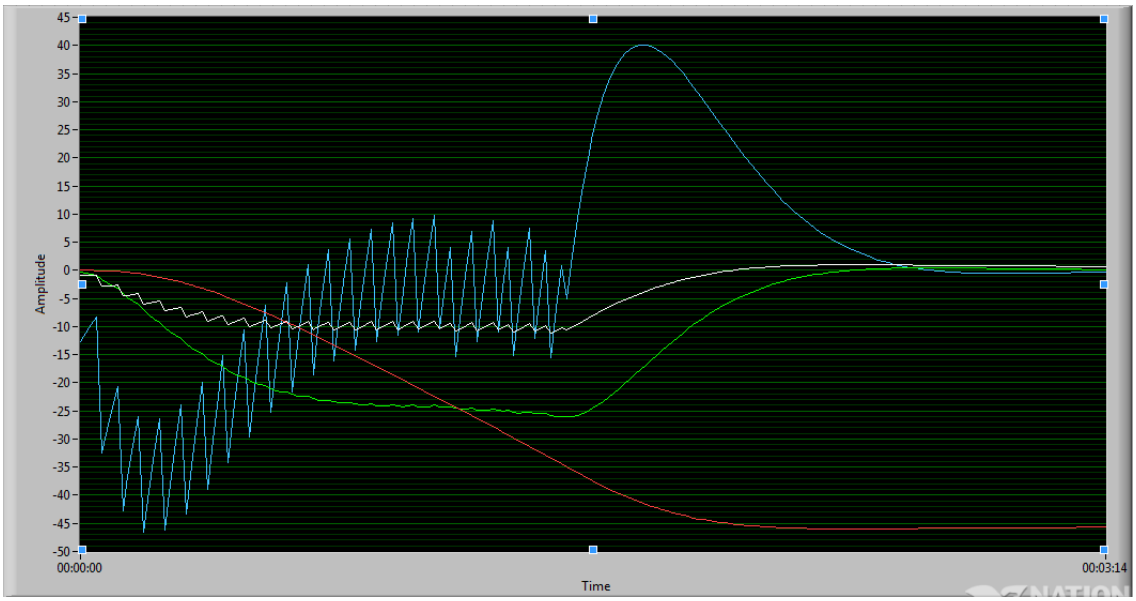

FIGURA 81 - GRÁFICO DE DADOS SOBRE MOTOR 2 (TETA 2 = - 45^{O})

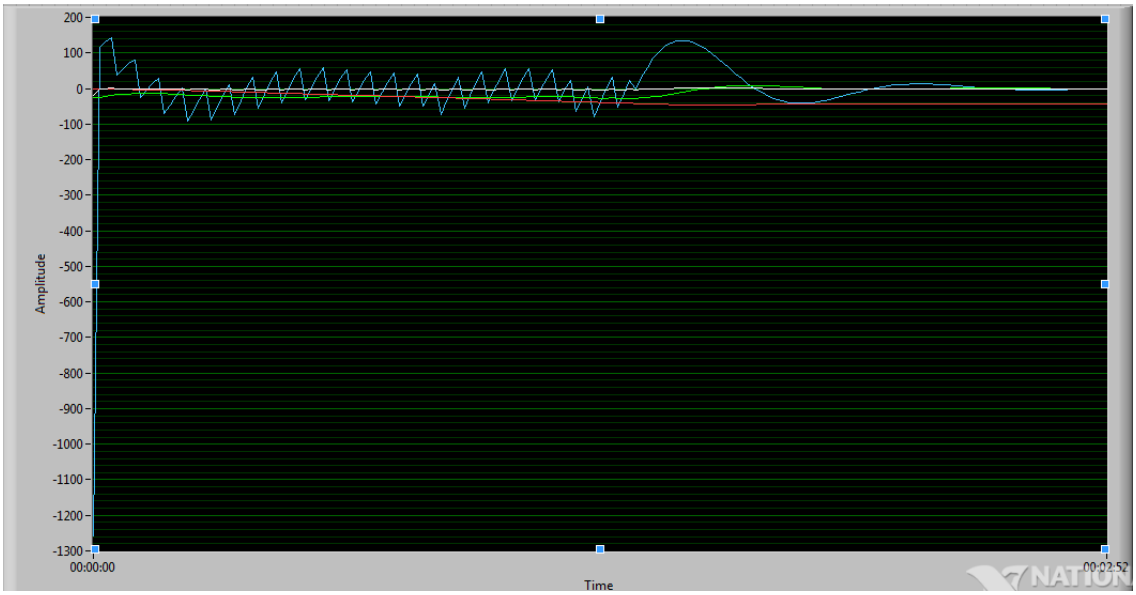

FIGURA 82 - GRÁFICO DE DADOS SOBRE MOTOR 3 (TETA 3 = - 45^{O})

Legenda:

Linha Branca – Sinal de feedback medido à entrada do compensador;

Linha Vermelha – Posição angular do motor;

Linha Verde – Velocidade angular do motor;

Linha Azul – Aceleração Angular do motor;

Nota: As imagens a cima apresentadas são referentes a dados obtidos directamente na malha de controlo e não no interface da aplicação para que se pudesse ter uma visão mais real do comportamento da malha de controlo. Isto acontece pois a interface amostra a malha de controlo com um período de 500 milissegundos.

Capítulo 6

6. Conclusões e trabalho futuro

Reflectindo sobre o trabalho realizado, as maiores dificuldades encontradas na construção da aplicação de simulação prenderam-se com a concepção e integração de uma malha de controlo capaz de simular o comportamento real dos vários motores do manipulador, ou seja, capaz de dotar o manipulador de dinâmica. Para agravar este problema contribuiu também a capacidade de processamento requerida para o bom funcionamento de toda a aplicação ser ela já elevada. Desta maneira foi necessário proceder-se a algum "trabalho de sapa" de modo a tornar a aplicação o mais eficiente possível. A necessidade deste "trabalho de sapa" tornou inviável a concepção de um segundo tipo de malha de controlo baseada na velocidade angular.

Em relação à aplicação de simulação aconselha-se como trabalho futuro a separação do bloco da malha de simulação da aplicação desenvolvida. Se se colocar num computador toda a parte de interface e bloco gerador de trajectórias e num segundo computador a malha de controlo e o software *RoboWorks* a carga a que é sujeito o processador do primeiro computador será substancialmente reduzida. A ligar os dois computadores seria usado um qualquer protocolo de comunicações.

Outra abordagem também poderia ser feita programando uma aplicação em tudo semelhante à que foi feita, contudo a malha de controlo a implementar seria diferente. A malha de controlo a implementar iria receber de um segundo computador (através de uma rede informática) o valor da posição, velocidade e aceleração angular do motor. Estes valores seriam processados e seria calculado um sinal de comando a enviar para o segundo computador. Desta maneira no segundo computador seria

necessário programar uma aplicação capaz de simular o funcionamento de um servo drive e motor. Essa aplicação teria que ser também devidamente sincronizada com a aplicação a correr no primeiro computador. Utilizando esta abordagem a aplicação a correr no primeiro computador desconheceria por completo se os dados que lhe eram enviados eram simulados ou se seriam oriundos de equipamento real. Isto permitira que a aplicação fosse posteriormente utilizada num manipulador real.

Finalmente, refiro que a natural progressão do trabalho desenvolvido seria a sua utilização no manipulador da empresa Selmatron, bastando para tal substituir o manipulador virtual criado em ambiente de simulação pelo manipulador real. Para tal seria necessário, porém, montar e testar o equipamento que esteve em falta durante o período de tempo em que decorreram os trabalhos desta dissertação. Só depois seria necessário efectuar a ligação à aplicação de software desenvolvida e aqui apresentada.

ANEXO I

MOTOR 1

Modelo: D314 – 037 A

Tipo: G L10

M_0 [Nm]: 2,7

I_0[A]: 6,7

U_d[V]: 310

N_n[nim^{-1}]: 4900

J[Kg cm^2]: 1,58

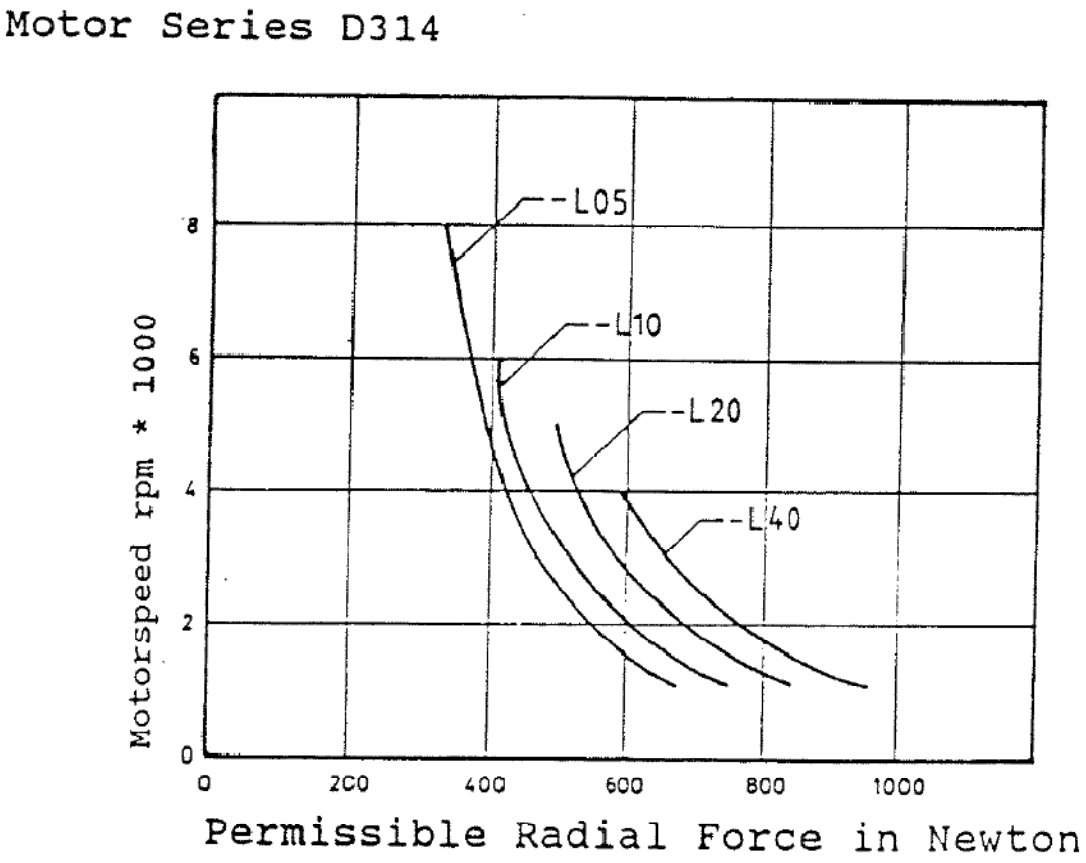

FIGURA 83 - GRÁFICO DA VARIAÇÃO DA CARGA IMPOSTA AO VEIO DO MOTOR EM FUNÇÃO DA VELOCIDADE DE ROTAÇÃO DO MOTOR DA FAMÍLIA D314

MOTOR 2

Modelo: D314 – 038A

Tipo: G L20

M_0 [Nm]: 5,0

I_0[A]: 9,3

U_d[V]: 310

N_n[nim^{-1}]: 3800

J[Kg cm^2]: 3,25

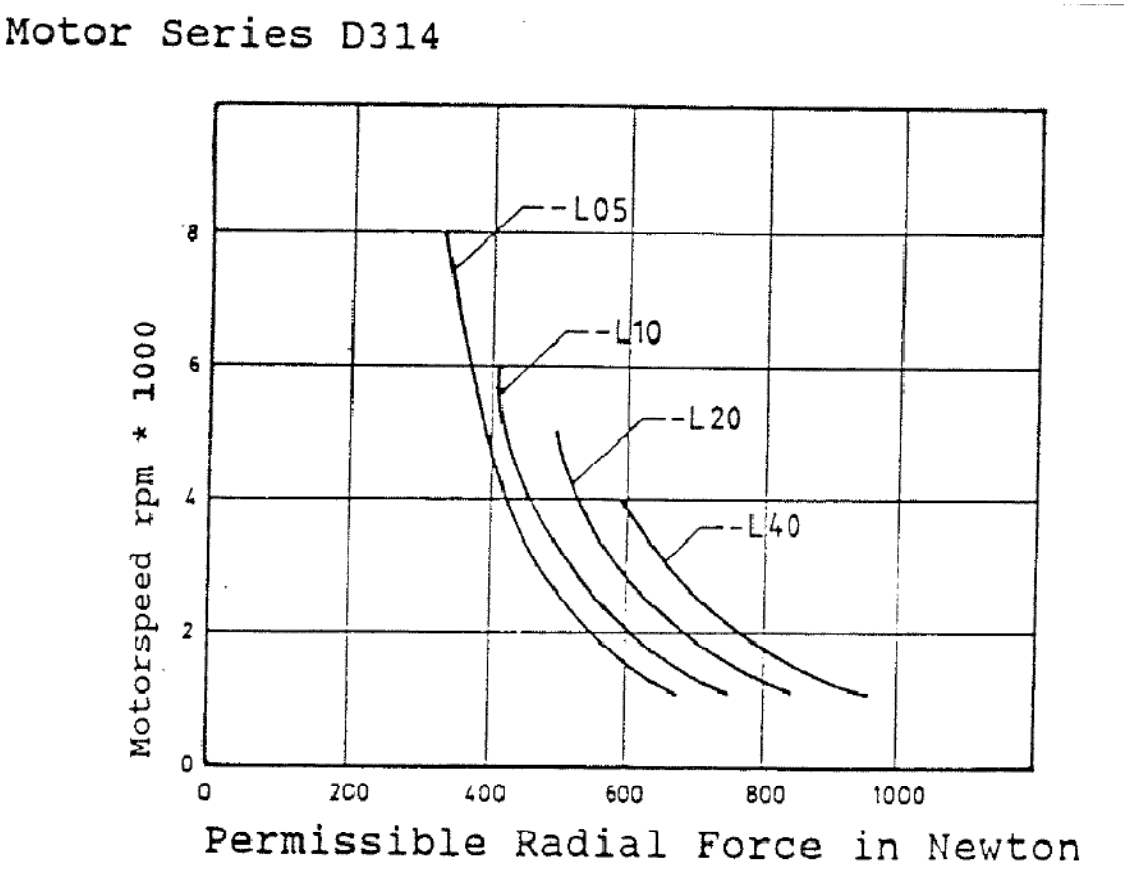

FIGURA 84 - GRÁFICO DA VARIAÇÃO DA CARGA IMPOSTA AO VEIO DO MOTOR EM FUNÇÃO DA VELOCIDADE DE ROTAÇÃO DO MOTOR DA FAMÍLIA D314

MOTOR 3

Modelo: D313 – 045A

Tipo: G L25

M_0 [Nm]: 2,3

I_0[A]: 4,8

U_d[V]: 310

N_n[nim^{-1}]: 4200

J[Kg cm^2]: 0,73

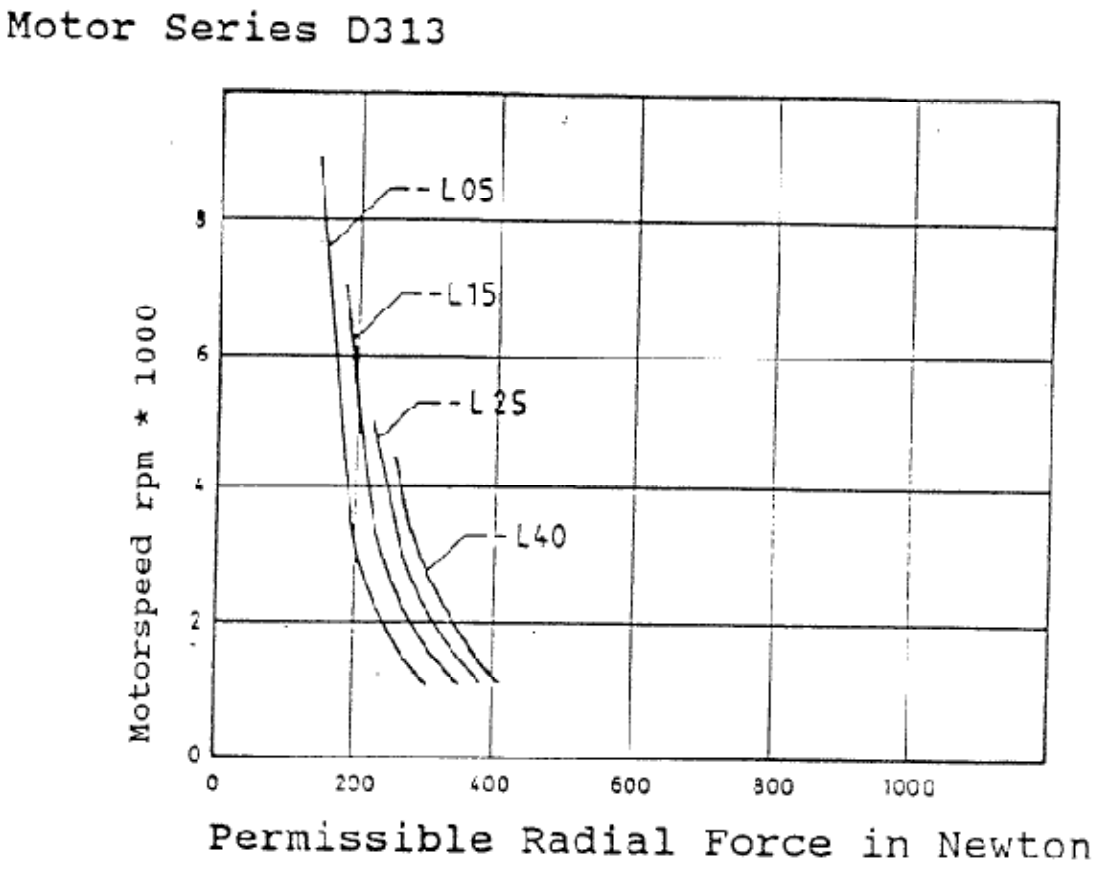

FIGURA 85 - GRÁFICO DA VARIAÇÃO DA CARGA IMPOSTA AO VEIO DO MOTOR EM FUNÇÃO DA VELOCIDADE DE ROTAÇÃO DO MOTOR DA FAMÍLIA D313

ANEXO II

Formulas trigonométricas usadas para realizar simplificações no calcula da cinemática inversa

$$sen(A \pm B) = sen(A).\cos(B) \pm \cos(A).sen(B)$$

$$\cos(A \pm B) = \cos(A).\cos(B) \mp sen(A).sen(B)$$

$$sen^2(A) + \cos^2(A) = 1$$

$$\cos(A + \frac{\pi}{2}) = -sen(A)$$

$$sen(A + \frac{\pi}{2}) = \cos(A)$$

Simplificação geométrica feita no cálculo de Teta 3

$$k_1.\cos(\phi) + k_2.sen(\phi) = k_3$$

$$\Rightarrow \phi = 2.a\tan(k_2 \pm \frac{\sqrt{k_1^2 + k_2^2 - k_3^2}}{k_1 + k_3})$$

Bibliografia:

[1] - PIRES, J. Norberto. “Automação Industrial”. Lisboa: ETEP, 2007

[2] – MCKERROW, P. “Introduction to Robotics”. Addison -Wesley, 1993

[3] - SANTOS, Vitor M. F. “Robótica Industrial”. Aveiro: Universidade de Aveiro, 2003

[4] - Wikipédia: “CANopen” - http://en.wikipedia.org/wiki/CANopen, consultado pela última vez a 30 de Outubro de 2009;

[5] - Wikipédia: “*EtherCAT*” - http://en.wikipedia.org/wiki/EtherCAT, consultado pela última vez a 2 de Novembro de 2009;

[6] – Wikipédia: “Profibus” - http://en.wikipedia.org/wiki/Profibus, consultado pela última vez a 2 de Novembro de 2009;

[7] - Wikipédia: “Servo drive” - http://en.wikipedia.org/wiki/Servo_drive, consultado pela última vez a 6 de Novembro de 2009;

[8] – Motor follower – Electronic Gearning, http://www.electricmotors.machinedesign.com/guiEdits/Content/bdeee5/bdeee5_5.aspx, consultado pela última vez a 6 de Novembro de 2009;

[9] – CRAIG, John J. “Introduction to robotics: mechanics and control” 2ª edição. Addison -Wesley, 1989;

[10] – FRANCISCO, António.”Motores Eléctricos”. Lisboa: ETEP – Edições Técnicas e Profissionais, 2008;

[11] – Wikipédia: “Termístor” - http://pt.wikipedia.org/wiki/Term%C3%ADstor,

consultado pela última vez a 10 Novembro de 2009;

[12] – *EtherCAT Techonology* - “*EtherCAT*” - http://www.ethercat.org/en/ethercat.html,

consultado pela última vez a 11 de Novembro de 2009;

[13] – *National Instrumentes – “EtherCAT” - http://zone.ni.com/devzone/cda/tut/p/id/7299*,

consultado pela última vez a 11 de Novembro de 2009;

[14] – SearchWinIT.com – "OCX" –

http://searchwinit.techtarget.com/sDefinition/0,,sid1_gci212689,00.html, consultado pela

última vez a 12 de Novembro de 2009;

[15] – Wikipédia: "*SERCOS interface*" - http://en.wikipedia.org/wiki/SERCOS_interface,

consultado pela última vez a 20 Outubro de 2009;

[16] – Sercos International Europe – "*SERCOS* III" - http://www.sercos.de/EUROPEzEnglish.15.0.html, consultado pela última vez a 14 de Setembro de 2009;

[17] – Newtonium: "RoboWorks" - http://www.newtonium.com/public_html/Products/RoboWorks/RoboWorks.htm, consultado pela última vez a 10 de Setembro de 2009;